Aula 08 (Prof. Diego Carvalho)

IFCE (Prof. Ciência da Computação-Metodologia e Técnicas da Comput) Conhecimentos - 2021(Pós-Edital)

Autor:

Diego Carvalho, Equipe Informática e TI, Evandro Dalla Vecchia Pereira , Pedro Henrique Chagas Freitas, Thiago Rodrigues

Cavalcanti, Raphael Henrique Lacerda

Aula 08 (Prof. Diego Carvalho)

20 de Setembro de 2021

Sumário

APRESENTAÇÃO

Queridos alunos, já peço desculpas de antemão. Essa é uma aula que não é difícil – pelo contrário, ela é bem tranquila. No entanto, não existe um entendimento comum entre os autores mais famosos sobre diversos tipos e classificações de testes. Se isso é ruim para o professor ministrar a aula, imagine para vocês que estão estudando. **Dito isso, fiquem calmos porque eu tentei deixar essa aula a mais didática possível.** *Fechou?* Então vem na fé...

PROFESSOR DIEGO CARVALHO - WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO

TESTES DE SOFTWARE

1 – Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Galera, existe uma frase que eu gosto muito que diz: "***A melhor maneira de construir confiança é tentar destruí-la***". Calma que eu não vou filosofar, isso é apenas um subterfúgio para que nós entendamos a importância de um teste. As imagens abaixo mostram testes de impacto automobilístico, que consistem no impacto de veículos automotores contra barreiras indeformáveis ou deformáveis.

Ela tem por objetivo avaliar a segurança automotiva para verificar se cumprem determinadas normas de segurança de proteção à colisão em situações de acidente de trânsito. Ora, quanto mais testes que buscam destruir o carro eu faço, mais confiança eu construo sobre o carro. *Por que?* **Porque, a cada teste realizado, eu identifico as falhas e as corrijo – sempre com o intuito de construir o carro mais seguro possível para os passageiros.**

Com testes de software, o processo é bastante similar! **O teste de software é o processo de executar um software com dois objetivos principais**: primeiro, demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos especificados; segundo, descobrir falhas ou defeitos no software que apresente comportamento incorreto, não desejável ou em não conformidade com sua especificação.

A primeira meta conduz ao Teste de Validação, no qual você espera que o sistema seja executado corretamente em um dado conjunto de casos de teste que refletem o uso esperado do sistema. A segunda meta conduz ao teste de defeitos, no qual são projetados casos de teste para expor defeitos. Os casos de teste podem ser obscuros e não precisam refletir como o sistema é usado normalmente.

Professor, testes resolvem todos os problemas de qualidade? Não! **Testes de Software não podem demonstrar – por exemplo – que um software é livre de defeitos ou que ele se comportará conforme especificado em todas as circunstâncias existentes**. É sempre possível que um teste

importante tenha sido ignorado. Já dizia Edsger Dijkstra: "*Os testes podem somente mostrar a presença de erros, não sua ausência*". *Captaram?*

A grande lance dos testes é convencer os desenvolvedores e os clientes do sistema de que o software é bom o suficiente para o uso operacional. O teste de software é um processo voltado a atingir a confiabilidade do software. Se um programa falhar frequentemente e repetidas vezes, pouco importa se outros fatores de qualidade de software são ou não aceitáveis. *Legal, mas qual é a definição de confiabilidade de software?*

A confiabilidade de software é definida em termos estatísticos como **a probabilidade de operação sem falhas de um programa de computador em um dado ambiente por um determinado tempo.** Diferentemente de outros fatores de qualidade, pode ser medida diretamente e estimada usando-se dados históricos e de desenvolvimento. Exemplo: estima-se que o programa X tenha uma confiabilidade de 0,999 depois de decorridas oito horas de processamento.

Em outras palavras, se o programa X tiver de que ser executado 1.000 vezes e precisar de um total de oito horas de tempo de processamento (tempo de execução), é provável que ele opere corretamente 999 vezes e falhe apenas uma vez! *Bacana?* Bem, para o Teste de Validação, um teste bem-sucedido é aquele em que o sistema funciona corretamente; para o Teste de Defeitos, **um teste bem-sucedido é o que expõe um defeito que causa o funcionamento incorreto.**

Já falamos tanto de testes de software, mas qual é a sua definição? Como é possível de prever, há diversas definições diferentes de diversos autores. **Myers, por exemplo, diz que é o *processo de executar um determinado software com a intenção de encontrar defeitos***. Já a IEEE 729 define como o *processo formal de avaliar um sistema ou componente por meios manuais ou automáticos para verificar se ele satisfaz os requisitos especificados*.

O famoso Glossário ISTQB (International Software Testing Qualifications Board) conceitua atividades do ciclo de vida, estáticas ou dinâmicas, voltadas para o planejamento, preparação e avaliação de produtos de software e produtos de trabalho relacionados **a fim de determinar se eles satisfazem os requisitos especificados e demonstrar que estão aptos para sua finalidade e detecção de defeitos**.

Interessante! *No entanto, o que define um bom teste?* Bem, existem quatro características que nos ajudam a classificar um teste como bom. São elas:

- **Característica:** *um bom teste tem alta probabilidade de encontrar defeitos.*

Para atingir esse objetivo de encontrar defeitos, **o testador deve entender o software e tentar desenvolver uma imagem mental de como o software pode falhar.** O ideal é que as classes de falhas sejam investigadas.

- **Característica:** *um bom teste não é redundante.*

O tempo e os recursos de teste são limitados. **Não tem sentido realizar um teste que tenha a mesma finalidade de outro teste.** Cada teste deve ter uma finalidade diferente (mesmo que seja sutilmente diferente).

- **Característica:** *um bom teste deverá ser "o melhor da raça".*

Em um grupo de testes com finalidades similares, as limitações de tempo e recursos podem induzir à execução de apenas um subconjunto desses testes. **Nesses casos, deverá ser usado o teste que tenha a maior probabilidade de revelar uma classe inteira de erros.**

- **Característica:** *um bom teste não deve ser nem muito simples nem muito complexo.*

Embora seja possível combinar algumas vezes uma série de testes em um caso de teste, os possíveis efeitos colaterais associados com essa abordagem podem mascarar erros. **Em geral, cada teste deve ser executado separadamente.**

Ao longo de diversos anos, a Engenharia de Software evoluiu bastante, de modo a sugerir alguns princípios que guiam os Testes de Software. Ao todo, são sete princípios fundamentais:

TESTES DEMONSTRAM A PRESENÇA DE DEFEITOS!

Um teste pode demonstrar a presença de defeitos, mas **não pode provar que eles não existem**. Ele reduz a probabilidade de que os defeitos permaneçam, mas mesmo se nenhum defeito for encontrado não quer dizer que ele não os tenha.

TESTES EXAUSTIVOS SÃO IMPOSSÍVEIS!

Testar todas as combinações de entradas e pré-condições é inviável, exceto para casos triviais. Em vez de realizar testes exaustivos, os riscos e prioridades são levados em consideração para dar foco aos esforços de teste.

TESTE O MAIS BREVE POSSÍVEL (ANTECIPADO!

Os defeitos encontrados nas fases iniciais do processo de desenvolvimento de software **são mais baratos de serem corrigidos do que aqueles encontrados já em fase produção**. Há, inclusive, técnicas de testes antes mesmo da implementação.

AGRUPEM OS DEFEITOS MAIS SENSÍVEIS!

Seguindo o Princípio de Pareto, 80% dos defeitos são causados por 20% do código. Ao identificar essas áreas sensíveis, **os testes podem priorizá-las, de forma a ter alta probabilidade de encontrar defeitos.**

PARADOXO DO PESTICIDA!

Caso os mesmos testes sejam aplicados repetidamente, em determinado momento eles deixam de ser úteis, ou seja, não conseguem encontrar nenhum novo defeito. **Por isso, os testes precisam ser revisitados com frequência.**

TESTES DEPENDEM DO CONTEXTO!

Os testes devem ser elaborados de acordo com o contexto de utilização do software. Ex: um sistema bancário deve ser testado de maneira diferente de uma rede social. Assim como testes de aplicação web têm foco diferente do desktop.

AUSÊNCIA DE DEFEITOS É UMA ILUSÃO!

Identificar e corrigir os problemas de um software não garantem que ele está pronto. *Os testes foram elaborados para identificar todas as possíveis falhas? O sistema atende às necessidades e expectativas dos usuários?* Logo, há outros fatores!

O objetivo do teste é encontrar erros, e um bom teste é aquele que tem alta probabilidade de encontrar um erro. Ao mesmo tempo, os próprios testes devem ter uma série de características que permitam atingir o objetivo de encontrar o maior número de erros com o mínimo de esforço. Dessa forma, um engenheiro de software deve projetar e implementar um sistema ou produto tendo em mente a *testabilidade. O que seria isso, Diego?*

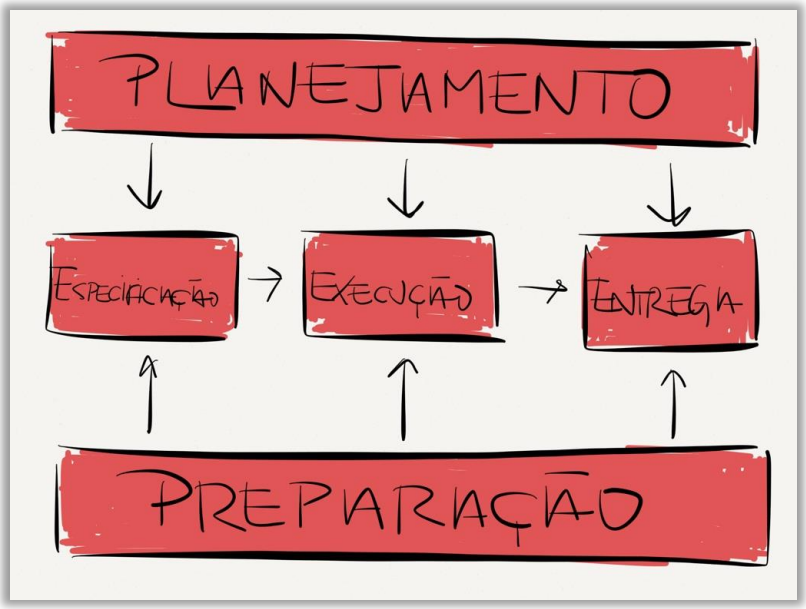
Cara, a testabilidade é definida como a facilidade com que um programa de computador pode ser testado. As seguintes características levam a um software testável:

CARACTERÍSTICA	DESCRIÇÃO
OPERABILIDADE	Grosso modo, podemos dizer que "*quanto melhor funciona, mais eficientemente pode ser testado*".
OBSERVABILIDADE	Grosso modo, podemos dizer que "*o que você vê é o que você testa*".
CONTROLABILIDADE	Grosso modo, podemos dizer que "*quanto melhor você pode controlar o software, mais o teste pode ser automatizado e otimizado*".
DECOMPONIBILIDADE	Grosso modo, podemos dizer que "*controlando o escopo, podemos isolar problemas mais rapidamente e realizar testes mais inteligentes*".
SIMPLICIDADE	Grosso modo, podemos dizer que "*quanto menos houver a testar, mais rapidamente podemos testá-lo*".
ESTABILIDADE	Grosso modo, podemos dizer que "*quanto menos modificações, menos interrupções no teste*".
COMPREENSIBILIDADE	Grosso modo, podemos dizer que "*quanto mais informações temos, de forma mais inteligente vamos testar*".

2 – Processo de Teste

INCIDÊNCIA EM PROVA: BAIXA

O processo de testes é caracterizado pela execução das principais etapas da atividade de testes e de suas subetapas. O objetivo dos processos de teste é direcionar e indicar a próxima atividade a ser feita. **Esses processos são conjuntos de boas práticas que orientam para chegar ao objetivo da melhor maneira e servem para minimizar os riscos causados por defeitos originados no processo de desenvolvimento.** Vejamos como é esse processo de testes...



- **Planejamento:** nesta etapa, elaboram-se o Projeto de Testes e o Plano de Testes. Ela acompanha todo o processo de teste, por meio de atividades como captação de requisitos, planejamento do projeto, análise de riscos e preparação de ambiente de testes.

- **Preparação:** nesta etapa, organiza-se o ambiente de testes (infraestrutura, equipamentos, hardware, software, pessoal capacitado, ferramentas e massa de testes adequadas) para que os testes sejam executados conforme planejados.

- **Especificação:** nesta etapa, temos as atividades de elaborar e revisar casos de testes e roteiros de testes (*scripts*). Esse último descreve a relação dos casos de testes e a previsão de execução dos testes.

- **Execução:** nesta etapa, os testes são executados conforme roteiros estabelecidos para os testes. Executa-se sempre que ocorrem mudanças na aplicação e analisam-se os testes executados com sucesso e os testes com defeito – os resultados obtidos são registrados.

- **Entrega:** nesta etapa, o projeto é finalizado, registra-se toda a documentação e relatam-se todas as incidências relevantes à melhoria do processo em um relatório de conformidades e não-conformidades – por fim, a documentação gerada é arquivada.

2.1 – Plano de Testes

INCIDÊNCIA EM PROVA: BAIXA

O processo de teste de software pode produzir diversos artefatos, dentre eles dois muito importantes: Plano de Testes e Casos de Testes. O Plano de Testes apresenta o planejamento para execução das atividades de testes, apresentando seu escopo, métodos empregados, prazo

estimado, nível de qualidade esperado, expecativa de capacidade, recursos que serão utilizados como ferramenta de apoio, e métricas e formas de acompanhamento do processo.

Como se trata de um documento gerencial, ele contempla a elaboração de um cronograma contendo todas as atividades e responsáveis por sua execução, podendo abranger casos de testes desde as primeiras entregas até o sistema completo. **Ele busca definir e comunicar a intenção do esforço de teste em determinada programação.** Como em outros documentos de planejamento, o principal objetivo é ganhar a aceitação e aprovação dos envolvidos no esforço de teste.

Para isso, o documento deve evitar informações que não serão compreendidas ou que serão consideradas irrelevantes pelos envolvidos. Planos de teste podem variar de organização para organização, mas em geral podem apresentar atributos apresentados na tabela a seguir. **Não é necessário decorar essas informações, basta entender que o Plano de Testes busca descrever toda informação que possa ser necessária para a execução das atividades de teste.**

SEÇÃO	CONTEÚDO
INTRODUÇÃO	Contém uma identificação do projeto, descrição dos objetivos do documento, o público ao qual ele se destina e escopo do projeto a ser desenvolvido.
REQUISITOS	Descreve em linhas gerais o conjunto de requisitos a serem testados no projeto a ser desenvolvido, comunicando o que deve ser verificado.
ESTRATÉGIAS E FERRAMENTAS	Apresenta um conjunto de tipos de testes a serem realizados, respectivas técnicas empregadas e critério de finalização de teste. Além disso, é listado o conjunto de ferramentas utilizadas.
EQUIPE E INFRAESTRUTURA	Contém descrição da equipe e da infraestrutura utilizada para o desenvolvimento das atividades de testes, incluindo: pessoal, equipamentos, software de apoio, materiais, etc.
CRONOGRAMA DE ATIVIDADES	Contém uma descrição de marcos importantes (*milestones*) das atividades (incluindo as datas de início e fim da atividade).
DOCUMENTAÇÃO COMPLEMENTAR	Apresenta-se uma relação dos documentos pertinentes ao projeto.

2.2 – Casos de Testes

INCIDÊNCIA EM PROVA: MÉDIA

O Caso de Teste é um artefato que contém um conjunto de condições e entradas utilizadas para testar um software. Em geral, possui os seguintes atributos: identificador, itens constantes no teste, especificação de entrada, especificação de saída, definição do ambiente necessário, necessidades especiais e dependências de outros casos de testes. Eu sei que esse assunto está muito abstrato para vocês, mas vamos tentar simplificar.

Imagine que uma fabricante de automóveis resolva revolucionar o mercado e construir um protótipo de carro voador. **Durante a construção, é necessário realizar diversos testes para que – ao final – o carro realmente possa voar sem problemas.** Logo, é chamada uma equipe de

testadores. Esses caras vão construir um plano de testes que vai descrever em linhas gerais toda informação que possa ser necessária para a execução das atividades de teste.

Após isso, é necessário descrever casos de teste, ou seja, um conjunto de condições e entradas utilizadas para testar o carro. Logo, esses casos de teste deverão apresentar um conjunto de ações e resultados esperados para elas, sendo que as ações são passos que serão executados pelo testador. Vejam abaixo possíveis exemplos – bastante simplificados – de casos de testes para o nosso carro voador hipotético.

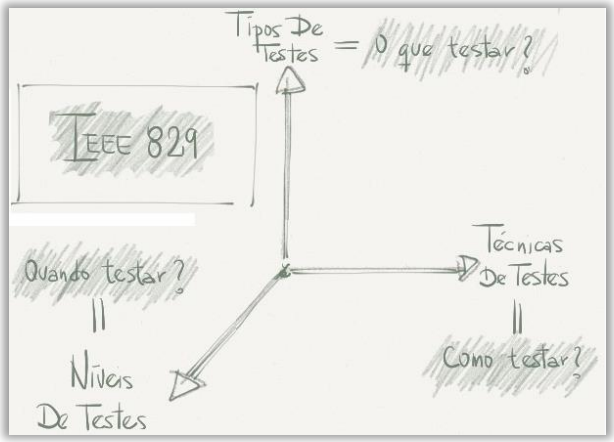
ENTRADA/CONDIÇÃO	RESULTADO ESPERADO
- Pressionar o botão de ligar	- Motor é acionado e o carro é ligado
- Desativar o freio de mão	- Desativar o sistema de frenagem
- Pressionar o pedal do acelerador	- Carro deverá começar sua aceleração vertical

É claro que isso é só um exemplo bobo, mas é legal para que vocês entendam que o testador realiza um conjunto de ações que devem gerar um resultado. **Caso o resultado não esteja de acordo com o esperado, significa que o testador encontrou uma falha – que é o principal objetivo de um teste** – por exemplo: o testador pressionou o botão de ligar do carro voador e o motor simplesmente não foi acionado.

No mundo do software, é bastante semelhante! Imagine um sistema de cadastro que exige que o usuário insira seu CPF. Um caso de teste óbvio é verificar se o campo de CPF obedece ao formato **###.###.###-##**. *Por que?* Porque esse é o formato padrão de um CPF! Logo, um caso de teste interessante seria tentar inserir o número **1234.56.7-7890** – se você conseguir, significa que o sistema está falhando, porque está aceitando um CPF em formato diferente do esperado.

ENTRADA/CONDIÇÃO	RESULTADO ESPERADO
- Validar a máscara do campo CPF	- Exibir máscara no formato ###.###.###-##
- Preencher CPF e clicar em salvar	- Registro salvo na tabela de cadastro
- Clicar no botão Voltar	- Sistema deve retornar à página inicial

Galera, nós podemos olhar para os testes por meio de três dimensões ou perspectivas diferentes: **Técnicas de Testes, Níveis de Testes e Tipos de Testes**. A imagem à esquerda as apresenta como um plano cartesiano de três dimensões – nós vamos ver nos próximos tópicos cada uma delas em detalhes. Na imagem à direita, podemos ver como essas dimensões de subdividem e qual é a função de cada uma delas: *como, quando e o que testar?*



TECNICA	NIVEL	TIPOS DE TESTES (LISTA EXEMPLIFICATIVA)		
CAIXA BRANCA	TESTE DE UNIDADE	SEGURANCA	VOLUME	INTEGRIDADE
		REGRESSAO	USABILIDADE	CARGA
CAIXA CINZA	TESTE DE INTEGRACAO	ESTRESSE	CONFIGURACAO	INSTALACAO
		FUMACA	MANUTENCAO	INTERFACE
CAIXA PRETA	TESTE DE SISTEMA	CENARIOS	COMPARACAO	RECUPERACAO
		COMPATIBILIDADE	ESCALABILIDADE	DOCUMENTACAO
	TESTE DE ACEITACAO	MIGRACAO	SCRIPT	MOBILE
		ETC	ETC	ETC
COMO TESTAR?	QUANDO TESTAR?	O QUE TESTAR?		

3 – Estratégia/Níveis de Testes

INCIDÊNCIA EM PROVA: MÉDIA

O software é testado para revelar erros cometidos inadvertidamente quando projetado e construído. *Mas como devemos conduzir os testes? Devemos estabelecer um plano formal para nossos testes? Devemos testar o programa como um todo ou executar testes somente em uma parte dele? Devemos refazer os testes quando acrescentamos novos componentes ao sistema? Quando devemos envolver o cliente?*

Essas e muitas outras questões são respondidas quando desenvolvemos uma estratégia de teste de software. *Por que ela é importante?* **O teste muitas vezes requer mais trabalho de projeto do que qualquer outra ação da engenharia de software.** Se for feito casualmente, perde-se tempo, fazem-se esforços desnecessários, e, ainda pior, erros passam sem ser detectados, portanto é razoável estabelecer uma estratégia sistemática e formal para teste de software.

De maneira genérica, podemos dizer que o teste começa pelo "pequeno" e passa para o "grande". Em outras palavras, os testes iniciais focalizam um único componente ou um pequeno grupo de componentes relacionados e aplicam-se testes para descobrir erros nos dados e na lógica de processamento que foram encapsulados pelo(s) componente(s). **Depois de testados, os componentes devem ser integrados até que o sistema completo esteja pronto.**

Nesse ponto, são executados muitos testes de ordem superior para descobrir erros ao atender aos requisitos do cliente. À medida que os erros forem descobertos, devem ser diagnosticados e corrigidos usando um processo chamado de depuração. Imaginem o motor de um carro: os testes começam nos componentes menores (Ex: vela de ignição, biela, virabrequim, pistão, válvula, etc) até terminar testando o funcionamento do carro como um todo.

Bem, muitas estratégias de teste de software já foram propostas na literatura. **Todas elas fornecem um modelo para o teste e todas têm as seguintes características genéricas:**

- Para executar um teste eficaz, proceder a revisões técnicas eficazes. Fazendo isso, muitos erros serão eliminados antes do começo do teste.

- O teste começa no nível de componente e progride em direção à integração do sistema computacional como um todo.

- Diferentes técnicas de teste são apropriadas para diferentes abordagens de engenharia de software e em diferentes pontos no tempo.

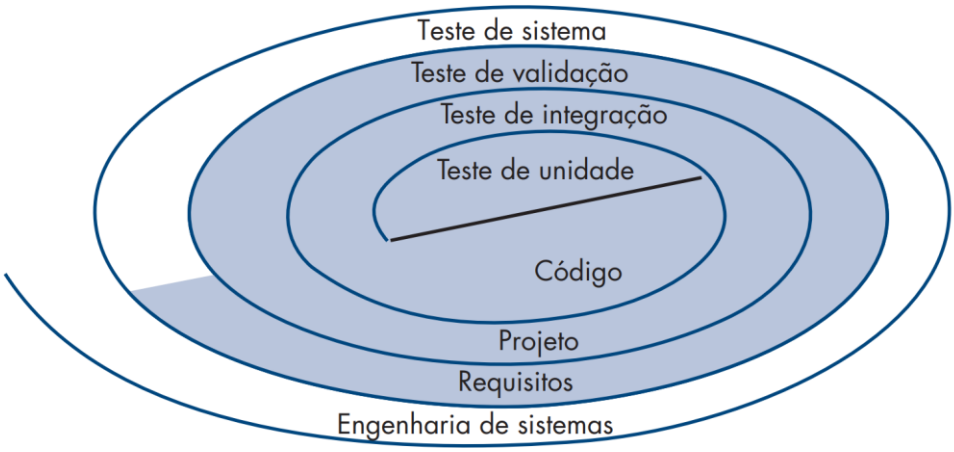
- O teste é feito pelo desenvolvedor do software e (para grandes projetos) por um grupo independente de teste.

- O teste e a depuração são atividades diferentes, mas a depuração deve ser associada com alguma estratégia de teste.

Nós sabemos que uma estratégia de teste de software deve acomodar testes de baixo nível, necessários para verificar se um pequeno segmento de código fonte foi implementado corretamente, bem como testes de alto nível, que validam as funções principais do sistema de acordo com os requisitos do cliente – como no exemplo do motor. **No entanto, existem abordagens diferentes no mundo do teste de software para executar uma estratégia.**

Em um dos extremos, pode-se esperar até que o sistema esteja totalmente construído e, então, executar os testes apenas no sistema completo esperando encontrar os erros. **Essa abordagem, embora atraente, simplesmente não funciona – resultará em um software defeituoso que desagrada todos os que investiram nele**. No outro extremo, você pode executar testes diariamente, sempre que uma parte do sistema for construída.

Essa abordagem, embora menos atraente, pode ser muito eficaz! Uma estratégia que é preferida pela maioria das equipes de software está entre os dois extremos. **Ela assume uma visão incremental do teste, conforme é apresentado na espiral**. Ela pode ser compreendida sob dois pontos de vista diferentes: pela visão do Processo de Software, deve ser lida no sentido anti-horário; pela visão de Testes de Software, deve ser lida no sentido horário. *Como assim, Diego?*



Bem, se nós quisermos analisar sob o ponto de vista do Processo de Software, inicialmente realizam-se atividades de **Engenharia de Sistemas**, passamos à análise de **Requisitos**, em seguida desenvolve-se um **Projeto** e – finalmente – chegamos à **Codificação**. Logo, para desenvolver softwares, percorre-se a espiral no sentido anti-horário ao longo de linhas que indicam a diminuição do nível de abstração em cada volta, isto é, iniciamos em um nível alto de abstração (entendendo o sistema e o domínio de informação) e terminamos com o código em si.

Por outro lado, se nós quisermos analisar sob o ponto de vista de Estratégia de Testes, inicialmente realizam-se **Testes de Unidade**, que se concentram em cada unidade do software (Ex: componente, classe, entre outros) conforme implementado no código-fonte. O teste prossegue movendo-se em direção ao exterior da espiral, passando pelo **Teste de Integração**, em que o foco está no projeto e construção da arquitetura de software.

Na mesma direção, encontramos o **Teste de Validação**, em que requisitos estabelecidos como parte dos requisitos de modelagem são validados em relação ao software criado. Finalmente, chegamos ao **Teste do Sistema**, no qual o software e outros elementos são testados como um todo. Para testar um software de computador, percorre-se a espiral em direção ao seu exterior, no sentido horário, ao longo de linhas que indicam o escopo do teste a cada volta.

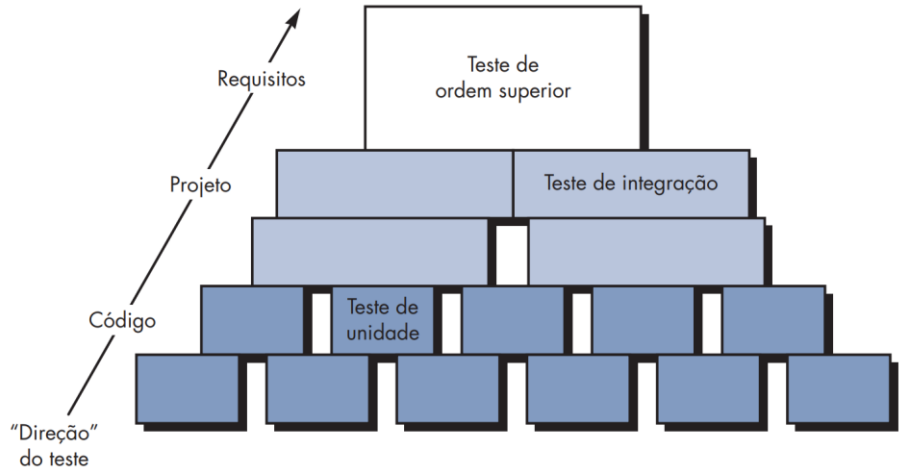
Considerando o processo de um ponto de vista procedimental, **o teste dentro do contexto de engenharia de software é na realidade uma série de quatro etapas que são implementadas sequencialmente – as etapas ilustradas na figura da página anterior.** Inicialmente, os testes focalizam cada componente individualmente, garantindo que ele funcione adequadamente como uma unidade, daí o nome teste de unidade.

O teste de unidade usa intensamente técnicas de teste com caminhos específicos na estrutura de controle de um componente para garantir a cobertura completa e a máxima detecção de erro. Em seguida, o componente deve ser montado ou integrado para formar o pacote completo de software. **O teste de integração cuida de problemas associados com aspectos de verificação e construção de programa.**

Técnicas de projeto de casos de teste que focalizam em entradas e saídas são mais predominantes durante a integração, embora técnicas que usam caminhos específicos de programa possam ser utilizadas para segurança dos principais caminhos de controle. **Depois que o software foi integrado (construído e montado), é executada uma série de testes de ordem superior.** Os critérios de validação – estabelecidos durante a análise de requisitos – devem ser avaliados.

O teste de validação proporciona a garantia final de que o software satisfaz a todos os requisitos informativos, funcionais, comportamentais e de desempenho. A última etapa de teste de ordem superior extrapola os limites da engenharia de software, entrando em um contexto mais amplo de engenharia de sistemas de computadores. O software, uma vez

validado, deve ser combinado com outros elementos do sistema (por exemplo: hardware, pessoas, base de dados).



O teste de sistema basicamente verifica se todos os elementos se combinam corretamente e se a função/desempenho global do sistema é conseguida. Há uma classificação que divide os testes em Testes de Baixo Nível (1º Nível) e Testes de Alto Nível (2º Nível). **No primeiro caso, o profissional deve ter um profundo conhecimento da estrutura interna do software – nesse caso, específico é comum que os testes sejam realizados pelo próprio desenvolvedor.** *Por que?*

Porque ele possui toda a carga de conhecimento que é necessária para realizar essas atividades. O primeiro nível é composto pelos Testes Unitários e Testes de Integração. **Já no segundo nível, não é necessário conhecimento da estrutura interna do software.** Os testes são guiados pelas especificações de negócio e pela lista de requisitos do software. O segundo nível é composto pelos Testes de Validação e Testes de Sistema.

3.1 – Teste de Unidade

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Também chamado de Teste de Componente/Módulo, focaliza o esforço de verificação na menor unidade de projeto do software: o componente ou módulo. A ideia aqui é testar caminhos importantes para descobrir erros dentro dos limites do módulo. Além disso, ele enfoca a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente, sendo possível ser conduzido em paralelo para diversos componentes.

A interface de um módulo é testada para assegurar que as informações fluam corretamente para dentro ou para fora da unidade de programa que está sendo testada. **Caminhos independentes da estrutura de controle são usados para assegurar que todas as instruções em um módulo tenham sido executadas pelo menos uma vez**. As condições-limite são testadas para garantir que o módulo opere adequadamente nas fronteiras estabelecidas para limitar o processamento.

Finalmente, são testados todos os caminhos de manipulação de erro. *Como assim, Diego?* Este é um processo de teste de defeitos e, portanto, sua meta é expor defeitos nesses componentes. *Professor, o que você quer dizer com componente ou módulo?* No código-fonte, seria uma função ou método individual de um objeto; classes de objeto com vários atributos e métodos; componentes compostos que constituem diferentes objetos ou funções.

Esses componentes compostos têm uma interface definida usada para acessar sua funcionalidade. **As funções ou métodos individuais são os tipos mais simples de componentes e seus testes são um conjunto de chamadas dessas rotinas com diferentes parâmetros de entrada**. Eles verificam o funcionamento de um pedaço do software isoladamente ou que possam ser testados separadamente.

O Teste de Unidade pode ser realizado antes ou depois que o código-fonte tenha sido escrito. **Geralmente são feitos pelos próprios desenvolvedores de maneira mais informal e, não, por especialistas em testes**. Por meio de Testes de Unidade, é possível encontrar problemas mais cedo; facilita mudanças; simplifica integrações; auxilia a documentação; melhora o projeto do software; entre outros. Vamos para as nossas tradicionais metáforas agora...

Vocês se lembram do nosso exemplo? Os componentes de um motor de carro seriam basicamente suas peças – elas são a menor unidade de um motor e, portanto, precisam ser testadas individualmente! **Além disso, é necessário testar as interfaces dessas peças, uma vez que elas serão integradas com outras peças futuramente como a biela e os pistões**. Com software, é bastante similar, mas são testadas unidades de código-fonte.


VIRABREQUIM


BIELA


PISTÃO

DEFINIÇÕES DE PROVA - TESTES DE UNIDADE
Testes de Unidade são aqueles realizados sobre as menores estruturas de código-fonte, como métodos e classes.
Testes de Unidade consistem em testar individualmente, componentes ou módulos de *software* que, posteriormente devem ser testados de maneira integrada.
Testes de Unidade focalizam cada componente de um software de forma individual, garantindo que o componente funciona adequadamente.
Testes de Unidade focalizam o esforço de verificação na menor unidade de projeto de software, isto é, no componente ou no módulo de software.
Testes de Unidade têm por objetivo explorar a menor unidade do projeto, procurando identificar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo separadamente.
Testes de Unidade enfocam a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente.
Testes de Unidade concentram o esforço de verificação na menor unidade de design de software.

Testes de Unidade concentram-se na lógica de processamento interno e nas estruturas de dados dentro dos limites de um componente.
Testes de Unidade têm por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente.
Testes de Unidade têm como foco as menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes.

3.2 – Teste de Integração

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Um novato no mundo do software pode levantar uma questão aparentemente legítima quando todos os módulos tiverem passado pelo teste de unidade: *ora, se todos módulos já foram testados e estão funcionando perfeitamente de forma individual, porque eles não funcionariam em conjunto?* Cara, o problema é justamente colocá-los todos juntos porque podem haver diversos problemas de interfaces.

No contexto de softwares, dados podem ser perdidos através de uma interface; um componente pode ter um efeito inesperado ou adverso sobre outro; subfunções, quando combinadas, podem não produzir a função principal desejada; imprecisão aceitável individualmente pode ser amplificada em níveis não aceitáveis; estruturas de dados globais podem apresentar problemas. Infelizmente, essa lista não tem fim...

O Teste de Integração é uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces. O objetivo é construir uma estrutura de programa determinada pelo projeto a partir de componentes testados em nível de unidade. **Muitas vezes, há uma tendência de tentar integração não incremental** – também chamado de abordagem Big Bang.

Nesse caso, todos os componentes são combinados com antecedência. **O programa inteiro é testado como um todo**. *Essa é uma boa abordagem?* Não, usualmente o resultado é o caos, porque muitos erros podem ser encontrados de uma só vez e a correção se torna complexa, visto que há muito espaço para procurar e isolar a causa do erro. Uma vez corrigidos esses erros, novos erros aparecem e o processo parece não ter fim.

Temos também a integração incremental, que é o oposto da abordagem Big Bang. O programa é construído e testado em pequenos incrementos ainda no ambiente de desenvolvimento, em que os erros são mais fáceis de isolar e corrigir; as interfaces têm maior probabilidade de serem testadas completamente; e uma abordagem sistemática de teste pode ser aplicada. *Professor, você está falando com uma linguagem muito complexa – você pode simplificar?* Claro...

Vamos pensar nas peças do nosso motor! Nós precisamos integrar o virabrequim, a biela e o pistão porque eles – juntos – ajudam a transformar o deslocamento em movimento de rotação. Em outras palavras, esse é o conjunto responsável por transmitir a energia gerada pelo propulsor para a transmissão, que a distribui para as rodas do veículo. **As interfaces de cada módulo foram testadas na etapa anterior, agora é necessário testar se elas se comunicam corretamente.** Vejam como as três peças mencionadas nos testes de unidade se integram.

DEFINIÇÕES DE PROVA - TESTES DE INTEGRAÇÃO
Testes de Integração são caracterizados por testar as interfaces entre os componentes ou interações de diferentes partes de um sistema.
Testes de Integração têm por objetivo verificar se as funcionalidades dos módulos testados atendem aos requisitos.
Testes de Integração visam testar as falhas decorrentes da integração dos módulos do sistema.
Testes de Integração são uma técnica sistemática para construir a arquitetura do software, enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces.
Testes de Integração têm por objetivo construir uma estrutura de programa determinada pelo projeto a partir de componentes já testados.
Testes de Integração são uma técnica utilizada para descobrir erros associados às interfaces na qual, a partir de componentes testados individualmente, se constrói uma estrutura de programa determinada pelo projeto.
Testes de Integração verificam o funcionamento em conjunto dos componentes do sistema, se são chamados corretamente e se a transferência de dados acontece no tempo correto, por meio de suas interfaces.
Testes de Integração verificam se os componentes do sistema, juntos, trabalham conforme descrito nas especificações do sistema e do projeto do programa.
Testes de Integração são uma técnica sistemática para construir a arquitetura do *software* enquanto conduz testes para descobrir erros associados às interfaces.

3.3 – Teste de Validação

INCIDÊNCIA EM PROVA: MÉDIA

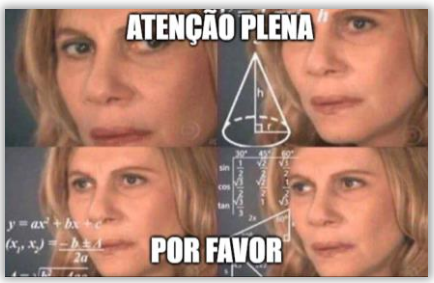
Também chamado de Teste de Aceitação, ele começa quando termina o teste de integração, quando os componentes individuais já foram exercitados, o software está completamente montado como um pacote e os erros de interface já foram descobertos e corrigidos. **Esse teste focaliza simplesmente em ações visíveis ao usuário e também em saídas do sistema reconhecíveis pelo usuário**. Se o usuário não vê ou reconhece, não é testado aqui!

A validação pode ser definida de várias maneiras, mas uma definição simples (embora rigorosa) afirma que **a validação tem sucesso quando o software funciona de uma maneira que pode ser razoavelmente esperada pelo cliente**. Nesse ponto, um desenvolvedor de software veterano pode dizer: *quem ou o que é o árbitro para decidir o que são expectativas razoáveis? Que critério é esse?* Difícil decidir isso...

Bem, se durante o processo de desenvolvimento foi desenvolvido um documento de requisitos de software, ela provavelmente descreverá todos os atributos do software visíveis ao usuário e conterá uma seção denominada Critérios de Validação, que ajuda a avaliar se os requisitos estão de acordo com as expectativas do usuário. **Para tal, existem os Testes Alfa e Testes Beta, que serão vistos posteriormente com mais detalhes**.

Para dar uma pequena noção sobre esses dois testes: o primeiro é conduzido por uma equipe de testadores no local em que ocorre o desenvolvimento; já o segundo é conduzido pelos próprios usuários no local em que ele realmente será utilizado. **De todo modo, a ideia por trás dos testes de validação/aceitação é funcionar como um teste formal sobre as necessidades dos usuários para determinar se o software satisfaz ou não os critérios de validação predefinidos.**

Vocês se lembram do nosso exemplo do motor? Pois é, vamos supor que aquele motor tenha sido encomendado para BMW para ser instalado em um carro da Citröen. Esse é o momento de chamar os funcionários da Citröen e testar o motor na presença deles, e também é o momento de entregar o motor para os próprios funcionários da Citröen testarem se está tudo certo. **O teste é também realizado em relação ao que estava especificado nos critérios de validação.**



Galera, nós sabemos que a verificação ocorre em relação à especificação de requisitos e a validação ocorre em relação às expectativas dos clientes. No entanto, a expectativa dos clientes é arbitrada em relação aos critérios de validação, que também estão presentes no documento de requisitos. **Logo, eu preciso da plena atenção de vocês agora porque as provas possuem algumas nuances em relação a esse assunto.**

Como os critérios de validação também pertencem ao documento de requisitos, **caso alguma questão afirme que a validação ocorre em relação aos requisitos não está errado!** *Diego, mas isso não faz o menor sentido!* Pois é, eu sei que isso é realmente confuso, mas nós temos que nos adaptar às bancas e, não, o contrário. Além disso, Roger Pressman também dá margem a esse entendimento em seu livro:

> *Como todas as outras etapas de teste, a validação tenta descobrir erros, mas o foco está no nível de requisitos — em coisas que ficarão imediatamente aparentes para o usuário final. A*

validação de software é conseguida por meio de uma série de testes que demonstram conformidade com os requisitos.

Vamos adotar um entendimento padrão? O teste de verificação tem por finalidade encontrar defeitos e inconsistências com relação a sua especificação de requisitos. Já o teste de validação ocorre em relação às expectativas do usuário e às funcionalidades percebíveis por ele. **Por outro lado, se uma questão afirmar que o teste de validação pode ocorrer em relação aos requisitos especificados, sem estar comparando com testes de verificação, recomendo marcar correto.**

DEFINIÇÕES DE PROVA - TESTES DE VALIDAÇÃO/ACEITAÇÃO
Testes de Validação focalizam ações e saídas, tais como percebidas pelo usuário final.
Testes de Validação são executados logo após montagem do pacote de software, quando os erros de interface já foram descobertos e corrigidos.
Testes de Validação têm como principal característica verificar o sistema em relação aos seus requisitos originais e às necessidades atuais do usuário.
Testes de Validação avaliam o software com respeito aos seus requisitos e detecta falhas nos requisitos e na interface com o usuário.

1

3.4 – Teste de Sistema

INCIDÊNCIA EM PROVA: MÉDIA

Galera, um software é apenas um elemento de um grande sistema de computador. Ao final do desenvolvimento, ele precisará ser incorporado aos outros elementos do sistema (por exemplo: hardware, pessoas, informações). Um problema clássico de teste de sistema é a "procura do culpado". **Isso ocorre quando um erro é descoberto e os desenvolvedores de diversos elementos do sistema começam a acusar um ao outro pelo problema.**

Em vez de adotar essa postura sem sentido, você deve se antecipar aos problemas potenciais de interface e realizar as seguintes atividades:

- Criar caminhos de manipulação de erro que testem todas as informações vindas de outros elementos do sistema;
- Executar uma série de testes que simulem dados incorretos ou outros erros potenciais na interface de software;
- Deve registrar os resultados dos testes para utilizar como evidência se ocorrer a caça ao culpado;
- Participar do planejamento e projeto de testes do sistema para assegurar que o software seja testado adequadamente.

O Teste de Sistema é na realidade uma série de diferentes testes cuja finalidade primária é exercitar totalmente o sistema. Embora cada um dos testes tenha uma finalidade diferente, todos funcionam no sentido de verificar se os elementos do sistema foram integrados adequadamente e executam as funções a eles alocadas. Galera, vamos finalizar as estratégias de testes pensando novamente em nosso exemplo.

Poxa, legal... eu testei os componentes do motor; depois eu testei como esses componentes se integravam; em seguida testei se aquilo satisfazia as necessidades dos clientes e os critérios de validação; e agora eu tenho que testar se aquele motor funciona corretamente integrado aos outros sistemas que integram o carro: sistema hidráulico, sistema elétrico, sistema de transmissão, sistema de arrefecimento, sistema de lubrificação, sistema de freios, entre outros.

Com software, é bastante similar! Um software é apenas parte de um sistema maior que envia e recebe dados e deve se integrar da melhor maneira possível. **O teste de integração verifica interfaces entre componentes do mesmo software, já o teste de sistema verifica interfaces entre componentes diferentes de um mesmo sistema** – incluindo softwares, hardwares, pessoas e informações, além dos requisitos funcionais e não-funcionais. *Capiche?*

DEFINIÇÕES DE PROVA - TESTES DE SISTEMA
Testes de Sistema incluem diversas modalidades de teste, cujo objetivo é testar o sistema computacional como um todo.
Testes de Sistema testam se o sistema cumpre seus requisitos funcionais e não funcionais.
Testes de Sistema avaliam o software com respeito ao seu projeto arquitetural e detecta falhas de especificação, desempenho, robustez e segurança.
Testes de Sistema visam a verificar o sistema, baseado em computador, não se limitando ao software, mas incluindo o processo como um todo, como hardware, pessoal e informação.

4 – Técnicas de Testes

As Técnicas se dividem em: Testes Caixa-Branca, Testes Caixa-Preta e Testes Caixa-Cinza. A primeira se foca nas estruturas internas dos procedimentos do sistema. A segunda se foca nas entradas e saídas especificadas nos requisitos funcionais. Por fim, a terceira se foca tanto em estruturas internas quanto nas entradas e saídas especificadas nos requisitos. Vejamos em detalhes cada uma delas. Venham comigo...

4.1 – Teste Caixa-Branca

INCIDÊNCIA EM PROVA: ALTÍSSIMA

A Técnica Caixa-Branca (também conhecida como Estrutural, Procedimental, Orientada à Lógica, Caixa-de-Vidro ou Caixa-Clara) analisa caminhos lógicos possíveis de serem executados, portanto é necessário ter conhecimento sobre o funcionamento interno dos componentes. Ela busca garantir que todos os caminhos independentes de um módulo sejam executados pelo menos uma vez.

Além disso, ele trata de todas as decisões lógicas para valores verdadeiros e falsos, além de executar laços dentro dos valores limites e avaliar as estruturas de dados internas do software – esse bando de termo técnica apenas mostra algumas formas de se avaliar a estrutura interna dos componentes. As técnicas principais são: Testes de Caminho Básico e Testes de Estruturas de Controle – não vamos entrar em detalhes sob 7 cada uma delas.

- **Caminho Básico:** permite derivar uma medida de complexidade lógica de um procedimento e a utiliza para definir um conjunto de caminhos de execução[1].

- **Estruturas de Controle:** permite validar estruturas de controle como estruturas de condição, estruturas de fluxo de dados ou estruturas de laços.

Galera, vocês se lembram que a estratégia de teste nos diz quando testar e a técnica de testes nos diz como testar? Se não, voltem algumas páginas! **Pois bem... a ideia da técnica de caixa-branca é testar a estrutura interna do software, isto é, seu código-fonte em si.** Ele é conhecido como caixa de vidro por uma razão: em uma caixa de vidro, eu consigo ver o que tem dentro; em uma caixa-preta, eu já não consigo fazer isso.

Então, a ideia aqui é: eu consigo ver as partes internas dos componentes, logo eu posso testá-las. Outro nome para essa técnica é teste estrutural! *Por que?* Porque eu consigo analisar a estrutura interna do código. Outro nome é teste procedimental! *Por que?* Porque eu consigo analisar como os procedimentos internos do componente. Outro nome é teste orientado à lógica! *Por que?* Porque eu consigo analisar sua lógica interna.

Vamos voltar ao nosso famoso motor de carro? **Fazendo uma comparação, o teste caixa-branca seria um teste em que o testador tem acesso às estruturas internas do motor.** Se ele quiser, ele pode abrir o motor para descobrir se o problema é, por exemplo, no virabrequim, biela ou pistão. Ele pode testar cada componente porque ele tem acesso a esses componentes e sabe como eles deveriam funcionar. Nós vamos ver no próximo tópico que nem sempre é assim...

[1] Geralmente é utilizado a Complexidade Ciclomática, que permite medir quantitativamente a complexidade lógica de um código por meio do número de caminhos independentes.

4.2 – Teste Caixa-Preta

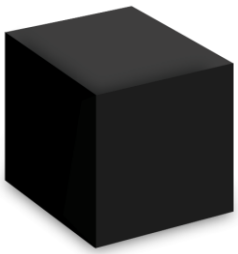
INCIDÊNCIA EM PROVA: ALTÍSSIMA

A Técnica Caixa-Preta (também conhecida como Comportamental, Funcional, Orientada a Dado, Orientada à Entrada/Saída ou Caixa-Escura) se baseia em pré-condições e pós-condições, geralmente sendo utilizada nas etapas posteriores da disciplina de testes. Ela busca funções incorretas ou inexistentes, erros de comportamento ou desempenho, erros de inicialização e interface, entre outros.

Deriva casos de teste a partir da especificação de requisitos, ignorando detalhes de implementação e se focando nas saídas geradas em resposta a entradas escolhidas e condições especificadas. Essa técnica tem o objetivo de verificar a funcionalidade e aderência aos requisitos, em uma ótica externa ou do usuário, baseado apenas em suas interfaces, sem se basear em qualquer conhecimento do código ( ) lógica interna do componente de software.

As técnicas principais são: Testes Baseados em Grafos, Particionamento de Equivalências, Análise de Valor Limite e Teste de Matriz Ortogonal.

- **Baseado em Grafos:** permite identificar os objetos e gera grafos para representá-los, testando-os e seus relacionamentos com o intuito de descobrir erros.
- **Partição de Equivalência:** permite agrupar os valores de entrada em categorias de dados para evitar redundância e aumentar a cobertura de testes do sistema.
- **Análise de Valor Limite:** permite exercitar os limites do domínio de entrada, tendo em vista que a maioria dos erros se encontram nas extremidades da entrada.
- **Matriz Ortogonal:** utilizado com entradas relativamente pequenas, casos de testes são espalhados uniformemente pelo domínio do teste para detectar falhas.



Galera, uma caixa-preta não permite que eu veja seus componentes internos. Logo, de forma diferente do teste caixa-branca, não é possível testar suas partes internas. *Ué, professor... então como eu vou testar?* Bem, nós temos um conjunto de pré-condições e pós-condições que também nos permitem encontrar erros e inconsistências. **Eu sei que, dadas condições específicas, eu devo alcançar resultados específicos.**

Outro nome para essa técnica é teste comportamental! *Por que?* Porque eu não consigo analisar a estrutura, mas eu consigo analisar seu comportamento esperado. Outro nome é teste Funcional! *Por que?* Porque eu consigo analisar como o software deveria funcionar. Outro nome é Orientado a Dado! *Por que?* **Porque eu não consigo analisar sua lógica, mas eu sei quais dados devem ser gerados como resultado** – além disso, dada uma entrada, temos uma saída específica.

Vamos voltar novamente ao nosso motor de carro? **Fazendo uma comparação, o teste caixa-preta seria um teste em que o testador não possui acesso às estruturas internas do motor.** Ele não pode abri-lo e verificar seu funcionamento interno. No entanto, ele sabe o comportamento que o motor deve ter quando se pressiona o acelerador, freio ou embreagem; quando se troca o tipo de combustível; quando se inicia a ignição; quando falta lubrificação; entre outros.

Para cada entrada ou pré-condição, há uma saída esperada. Dessa forma, é possível descobrir qual é o problema do motor. Galera, isso serve para tudo! Você não tem acesso às partes internas dos componentes que te fornecem internet em casa. **No entanto, você pode fazer testes caixa-preta! Se sua internet cair, você pode testar se o cabo de força está conectado na tomada; se o cabo de internet está conectado ao computador; entre outros.**

Como você sabe que, dadas entradas específicas, você deve ter saídas específicas, você pode realizar diversos testes para identificar problemas mesmo sem ter acesso a componentes internos. **Você não sabe como uma calculadora faz internamente para realizar um cálculo, mas você sabe que – ao calcular 3x7 – o resultado necessariamente terá que ser 21. Qualquer outro resultado implicaria em um defeito da calculadora!** *Bacana?* Fechou...

4.3 – Teste Caixa-Cinza

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

O Teste Caixa-Cinza é uma versão híbrida entre os Testes Caixa-Branca e os Testes Caixa-Preta. Nesse sentido, essa técnica analisa a parte lógica mais a funcionalidade do sistema, fazendo uma comparação do que foi especificado com o que está sendo realizado. Usando esse método, o testador comunica-se com o desenvolvedor para entender melhor o sistema e otimizar os casos de teste que serão realizados.

Isso envolve ter acesso a estruturas de dados e algoritmos do componente a fim de desenvolver os casos de teste, que são executados como na técnica da caixa-preta. Manipular entradas de dados e formatar a saída não é considerado caixa-cinza, pois a entrada e a saída estão claramente fora da caixa-preta. Alguns autores definem o Teste de Integração como um Teste Caixa-Cinza – esse raramente cai em prova, galera...

5 – Tipos de Testes

Galera, agora veremos vários e vários tipos de testes. É importante que vocês saibam que cada autor pode criar seu tipo de teste e eles – por vezes – se contradizem. **A ideia aqui é apresentar aqueles testes que possuem maior destaque entre os autores mais consagrados de engenharia de software, mas deixando claro que não se trata de uma lista exaustiva.** Eventualmente, vocês podem encontrar um teste que não foi mencionado em aula. *Animados?*

5.1 – Teste de Desempenho

INCIDÊNCIA EM PROVA: ALTA

Para alguns tipos de software, é inaceitável que um software execute suas funções, mas não esteja em conformidade com seus requisitos de desempenho. **O teste de desempenho (ou performance) é projetado para testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado.** O teste de desempenho é feito em todas as etapas no processo de teste e deve ser realizado, se possível, o quanto antes.

Até mesmo em nível de unidade, o desempenho de um módulo individual pode ser avaliado durante o teste. No entanto, o verdadeiro desempenho de um sistema só pode ser avaliado depois que todos os elementos do sistema estiverem totalmente integrados. **Os testes de desempenho muitas vezes são acoplados ao teste de esforço e usualmente requerem instrumentação de hardware e software.** b

Em outras palavras, frequentemente é necessário medir a utilização dos recursos (por exemplo, ciclos de processador) de forma precisa. Instrumentação externa pode monitorar intervalos de execução, log de eventos (por exemplo, interrupções) à medida que ocorrem, e verificar os estados da máquina regularmente. **Monitorando o sistema com instrumentos, o testador pode descobrir situações que levam à degradação e possível falha do sistema.**

Eu sei, linguagem muito técnica! Vamos falar de maneira bem clara: teste de desempenho trata do esforço em assegurar que o sistema computacional pode operar sob a carga de operação especificada. Imaginem o site do Estratégia Concursos em dia de Black Friday! **Se eu não faço testes para avaliar qual a carga de acessos simultâneos ele é capaz de suportar, o site pode cair, perde clientes e perder dinheiro.** Logo, é um teste importantíssimo!

Esse teste pode ser usado também para identificar gargalos, determinar conformidades com os requisitos não-funcionais de desempenho e coletar outras informações, como hardware necessário para a operação da aplicação. Em geral, devem ser projetados para assegurar que o

sistema pode operar na carga especificada. **Isso envolve o planejamento de uma série de testes em que a carga é constantemente aumentada até que o desempenho se torne inaceitável.**

Galera, alguns autores consideram Testes de Stress e Testes de Carga como subgêneros dos Testes de Desempenho. Vejamos em detalhes...

5.1.1 – Teste de Estresse

INCIDÊNCIA EM PROVA: ALTA

Também chamado de teste de esforço, serve para colocar os programas em situações anormais. Essencialmente, o testador que executa teste por esforço pergunta: *até onde podemos forçar o sistema até que ele falhe?* Esse teste usa um sistema de maneira que demande recursos em quantidade, frequência ou volumes anormais. Podem ser realizados testes em condições de alta taxa de entrada de dados, máximo de memória ou processamento, entre outros.

Em suma, esse tipo de teste é uma forma de teste deliberadamente intenso e completo utilizado para determinar a estabilidade de um determinado sistema ou entidade. Envolve o teste para além da capacidade operacional normal, muitas vezes até um ponto de ruptura, a fim de observar os resultados. **Assim como o teste de desempenho – que está em uma hierarquia superior – o teste de estresse testa basicamente requisitos não-funcionais.**

5.1.2 – Teste de Carga

INCIDÊNCIA EM PROVA: ALTA

Os Testes de Carga são a forma mais simples de testes para compreender o comportamento de um sistema sob uma carga específica. Ele é capaz de determinar o comportamento de um sistema sob condições especificadas ou acordadas. A carga pode ser, por exemplo, o número de usuários simultâneos esperados na aplicação. Em suma: testes de desempenho podem se dividir em teste de stress e teste de carga.

Os testes de carga procuram determinar como o software responderá a várias condições de carga (Ex: número de usuários, número de transações por usuário por unidade de tempo, carga de dados processados por transação) de acordo com os limites de operação do sistema. Já o teste de estresse ultrapassa os limites operacionais do sistema para determinar com que capacidade o ambiente pode lidar. *Fechou?*

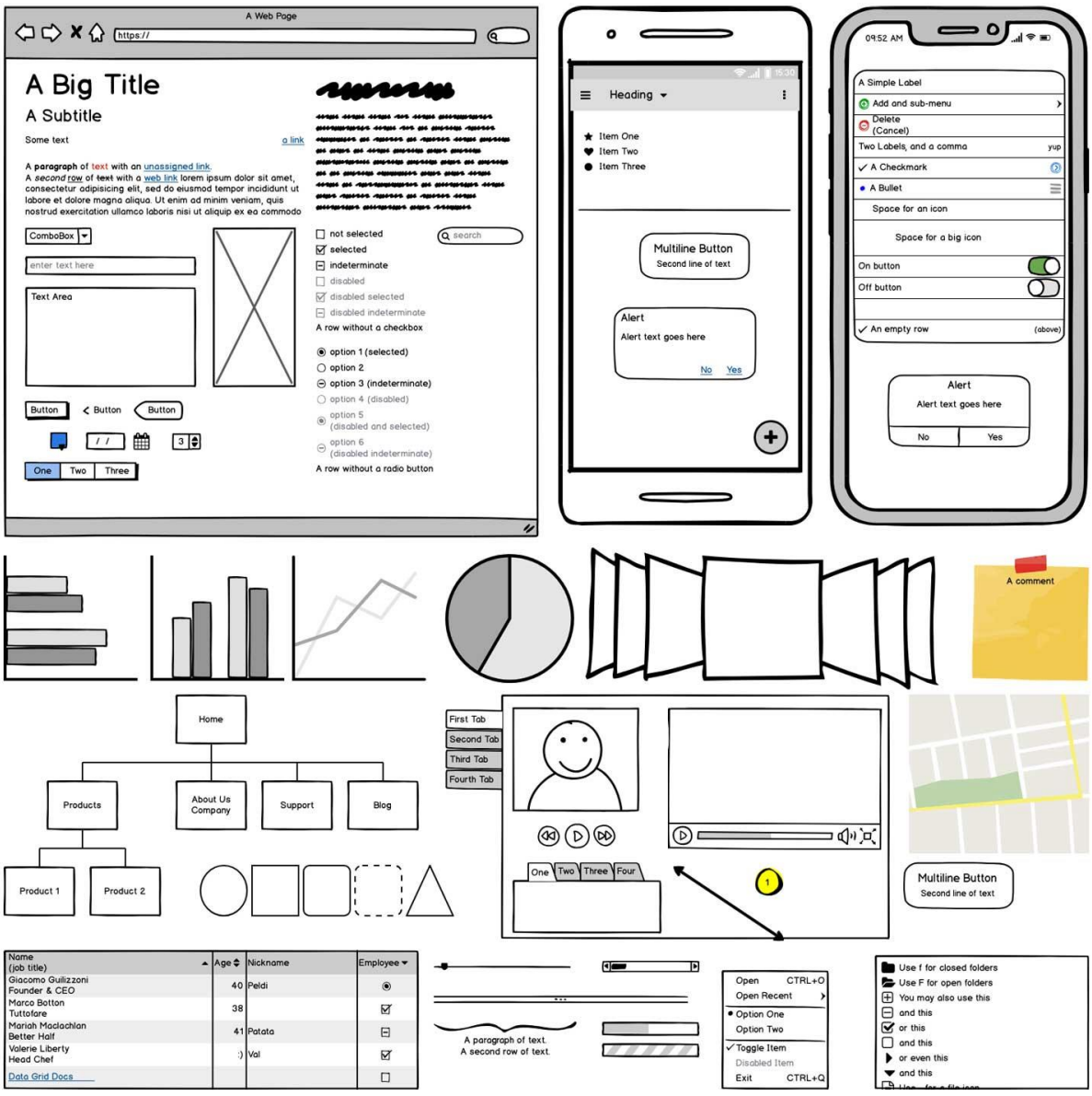
5.2 – Teste de Usabilidade

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Testes de Usabilidade avaliam o grau com o qual os usuários podem interagir efetivamente com o software e o grau em que o software dirige as ações do usuário, proporciona uma boa realimentação e reforça uma abordagem de interação consistente. **Basicamente, os testes de usabilidade são projetados para determinar o grau com o qual a interface de um software facilita a vida do usuário.**

Dito de outra forma, esse teste trata do esforço em demonstrar falhas na facilidade de uso do software pelos usuários finais. Evidentemente, ele enfatiza fatores humanos, interface gráfica, ajuda online, estética/layout, wizards, documentação do usuário, material de treinamento, acesso às funcionalidades, entre outros. **Costuma-se dizer que uma boa interface com o usuário deve ser fácil de usar e de entender.**

Em suma, trata-se de um teste focado na experiência do usuário que avalia a facilidade de uso e aprendizado de um software e o desempenho da interação homem-computador com o intuito de obter indícios do nível de satisfação do usuário. Para tal, esse teste se utiliza de diversas técnicas, como a utilização de protótipos para abordar a interatividade, layout, clareza, estética, características da tela, sensibilidade ao tempo, personalização, acessibilidade, etc.



EXEMPLO DE SOFTWARE QUE PERMITE A CONSTRUÇÃO DE PROTÓTIPOS DE TELA (BALSAMIQ)

5.3 – Teste de Regressão

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Galera, pensem lá no teste de integração. *Como ele funciona?* Ele testa como os componentes se comportam quando estão integrados com outros componentes. **Cada vez que um novo módulo é acrescentado como parte do teste de integração, o software muda.** Novos caminhos de fluxo de dados são estabelecidos, podem ocorrer novas entradas e saídas, e uma nova lógica de controle pode ser invocada.

Essas alterações podem causar problemas com funções que antes funcionavam corretamente. **No contexto de uma estratégia de teste de integração, o teste de regressão é a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados.** Sim, pessoal... por vezes, você vai corrigir um problema, obtém êxito, mas acaba inserindo defeitos em outro local.

Em um contexto mais amplo, testes bem-sucedidos (de qualquer tipo) resultam na descoberta de erros, e os erros devem ser corrigidos. Sempre que o software é corrigido, algum aspecto da configuração do software (o programa, sua documentação, ou os dados que o suportam) é alterado. **O teste de regressão ajuda a garantir que as alterações (devido ao teste ou por outras razões) não introduzam comportamento indesejado ou erros adicionais.**

O teste de regressão pode ser executado manualmente, reexecutando um subconjunto de todos os casos de teste ou usando ferramentas automáticas de captura/reexecução. Ferramentas de captura/reexecução permitem que o engenheiro de software capture casos de teste e resultados para reexecução e comparação subsequente. À medida que o teste de integração progride, o número de testes de regressão pode crescer muito.

Dessa forma, o conjunto de testes de regressão deve ser projetado de forma a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. **É impraticável e ineficiente reexecutar todos os testes para todas as funções do programa quando ocorre uma alteração.** *Entendido?* Galera, esse é talvez o tipo de teste mais recorrente em prova, então vamos resumir tudo que vimos...

O teste de regressão busca verificar a existência de defeitos após alterações em um sistema já testado. A ideia é descobrir erros por meio da reaplicação dos testes a um programa por conta de modificações ou novas funcionalidades para garantir que não foram introduzidos novos defeitos em componentes já testados anteriormente. **Eles são realizados geralmente durante a fase de manutenção, para mostrar que as modificações efetuadas estão corretas.** Como é na prática...

Imagine que eu acabo de desenvolver o novo site do Estratégia Concursos! Antes de disponibilizar para os alunos, eu faço vinte testes diferentes. Todos são exitosos, então eu finalmente lanço a versão 1.0 e disponibilizo para os alunos. **Só que aí o Prof. Ricardo Vale me**

pede para inserir uma nova funcionalidade que os alunos estão pedindo desesperadamente. Eu implemento essa nova funcionalidade e lanço a versão 1.1!

Maaaaaaaaaaaas... antes de lançar essa nova versão, eu faço um teste de regressão. *Por que, Diego?* **Porque essa nova funcionalidade que eu implementei pode ter causado algum problema naquilo que já estava funcionando perfeitamente na versão anterior.** *Então, o que eu faço?* Eu reaplico os vinte testes diferentes da versão anterior para garantir que essa nova funcionalidade não introduziu novos defeitos que comprometam as funcionalidades da versão anterior.

5.4 – Teste de Fumaça

INCIDÊNCIA EM PROVA: BAIXA

Teste fumaça é uma abordagem de teste de integração usada frequentemente quando produtos de software são desenvolvidos. É projetado como um mecanismo de marcapasso para projetos com prazo crítico, permitindo que a equipe de software avalie o projeto frequentemente. Uma série de testes é criada para expor erros que impedem a construção de executar corretamente sua função.

A finalidade deverá ser descobrir erros bloqueadores ou impeditivos (*showstopper*) que apresentam a mais alta probabilidade de atrasar o cronograma do software. O teste fumaça pode ser caracterizado como uma estratégia de integração rolante. O software é recriado (com novos componentes acrescentados) e o teste fumaça é realizado todos os dias. O teste fumaça deve usar o sistema inteiro de ponta a ponta.

Ele não precisa ser exaustivo, mas deve ser capaz de expor os principais problemas. O teste fumaça deve ser bastante rigoroso de forma que, se a construção passar, você pode assumir que ele é estável o suficiente para ser testado mais rigorosamente. *Diegão, essa linguagem está muito técnica, manda aquele exemplo maroto?* Claro, seus lindos! Deixe-me contar uma coisa muito comum no mundo dos programadores...

Por vezes, você passa meses desenvolvendo um novo software. Como você está seguro e confiante de tudo que tem desenvolvido, você nem sequer testa o que você fez e já chama o cliente para o qual você está desenvolvendo o software para apresentar sua obra de arte. O cliente traz toda a equipe dele para ver o novo software e você começa a sua apresentação. **São quinze pessoas esperando que você mostre o novo sistema e você mal começa a usar e... bug!**

Pior que isso! É um bug impeditivo, que te impossibilita de apresentar o restante do sistema. *Dá um exemplo, Diego?* Claro, você vai mostrar o sistema que você fez e ele dá um erro logo na hora de logar no sistema, ou seja, como você não consegue sequer logar, você não consegue apresentar nada para o seu cliente e a equipe de quinze pessoas. **Galera, juro para vocês, eu já vi isso acontecer tantas vezes... é frustrante, decepcionante e constrangedor.**

O teste de fumaça (*smoke testing*) é aquele teste que busca verificar se o software roda e provê suas funcionalidades e características básicas. **Prestem atenção: a ideia do teste de fumaça nem é ser detalhista como vários outros que nós vimos – é simplesmente ver se o software roda e se ele faz o básico, fundamental, o caminho feliz do software.** Caso ele passe nesse teste, ele estará habilitado a receber testes mais detalhados posteriormente. *Bacana?*

5.5 – Teste de Comparação

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Também conhecidos como testes back-to-back, eles são bem comuns em sistemas críticos. Realizam-se testes em versões diferentes do software e os resultados são comparados – não só o conteúdo, mas também o tempo de resposta. Em outras palavras, versões diferentes recebem a mesma entrada e suas saídas são comparadas. Imagine a situação em que uma empresa já possua um sistema legado/antigo e encomendou o desenvolvimento de uma nova versão do sistema.

Essa nova versão é mais moderna e conta – por exemplo – com tecnologias mais novas, linguagens de programação diferentes ou novos sistemas de armazenamento. **Nesse contexto, o teste de comparação é útil para verificar se a nova versão do sistema implementa as mesmas funcionalidades e com os mesmos resultados do sistema antigo.** Ao final, é interessante gerar um relatório que deixem expostas possíveis diferenças.

Galera, é muito comum haver uma confusão entre teste de regressão e teste de comparação. O primeiro teste verifica se uma alteração no software não inseriu bugs em partes já previamente testadas. O segundo teste executa um mesmo teste em duas versões diferentes do sistema e compara os resultados obtidos. *Entendido?* Vamos ver agora o único exercício que eu encontrei sobre esse tipo de teste...

5.6 – Testes Alfa e Beta

INCIDÊNCIA EM PROVA: MÉDIA

É praticamente impossível para um desenvolvedor de software prever como o cliente realmente usará um programa. As instruções de uso podem ser mal interpretadas, combinações estranhas de dados podem ser usadas regularmente; resultados que pareciam claros para o testador podem ser confusos para um usuário. **Pois bem, existem alguns tipos de testes de aceitação que são utilizados para permitir ao cliente validar todos os requisitos: testes alfa e beta.**

Ambos são conduzidos pelo usuário final e, não, por testadores, programadores ou engenheiros de software – sendo que um teste de aceitação pode variar desde um informal *test drive* até uma série de testes planejados e sistematicamente executados. Na verdade, um teste de aceitação pode ser executado por um período de semanas ou meses, descobrindo assim erros cumulativos que poderiam degradar o sistema ao longo do tempo.

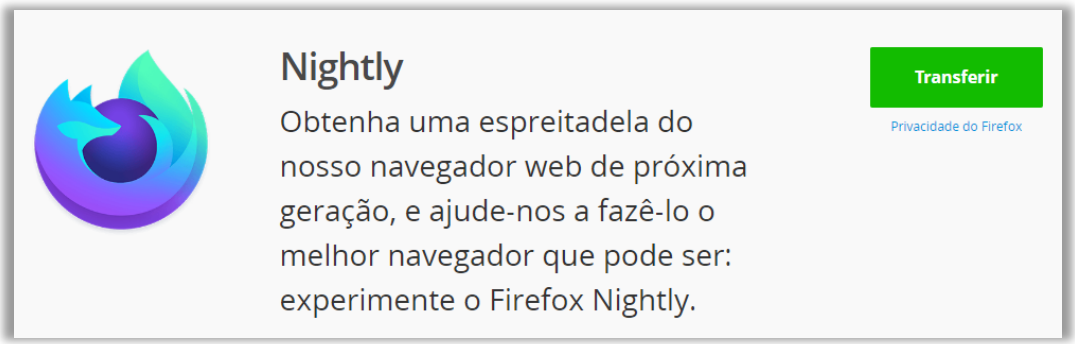
Se um software é desenvolvido como um produto para ser usado por muitos clientes, é impraticável executar testes formais de aceitação para cada cliente. **Muitos construtores de software usam os testes alfa e beta para descobrir erros que somente o usuário final parece ser capaz de encontrar.** Pois bem, então vamos descobrir agora a grande diferença entre os testes alfa e os testes beta.

Testes Alfa são testes de aceitação executados quando o desenvolvimento está próximo a sua conclusão. Este tipo de teste ocorre no ambiente do desenvolvedor e é geralmente realizado por um grupo representativo de clientes e usuários finais e, não, por programadores ou testadores. Em suma, eles são executados por usuários nas instalações do desenvolvedor do software, isto é, em um ambiente controlado. **Na prática, o programador fica olhando e registrando erros.**

Testes Beta são testes de aceitação executados quando o desenvolvimento está praticamente concluído e quando o maior número possível de defeitos precisa ser encontrado antes do lançamento do produto. Este tipo de teste também é realizado por clientes e usuários finais e, não, por programadores ou testadores. **Ademais, eles ocorrem nas instalações do usuário, isto é, no local real de utilização/trabalho – em um ambiente não controlado.**

Diferentemente do teste alfa, o desenvolvedor geralmente não está presente. Logo, o teste beta é uma aplicação "ao vivo" do software em um ambiente que não pode ser controlado pelo desenvolvedor. O cliente registra todos os problemas (reais ou imaginários) encontrados e relata para o desenvolvedor em intervalos regulares. **Os engenheiros de software, em geral, fazem modificações e então preparam a liberação do software para todos os clientes.**

Galera, uma dica! **Quando eu estudava para concursos, eu tinha grande dificuldade de memorizar qual era o Teste Alfa e o Beta.** *Como eu fiz para memorizar?* Bem, eu me lembrava dos aplicativos beta que eu instalava no meu computador. *Como é, Diegão?* Por exemplo: Firefox Nighly! Para quem não sabe, o Navegador Firefox possui várias versões (entrem no site e vejam vocês mesmos...).



Uma dessas versões é chamada Firefox Nightly e ela é disponibilizada antes da versão final para quem quiser testar, descobrir as próximas novidades e reportar eventuais erros de funcionalidades, compatibilidade, estabilidade, etc. Percebam: quem testa o software é o

usuário final em seu próprio computador. **No caso, eu no meu quarto – um ambiente não controlado pelos programadores.** Vejam abaixo o link para enviar um feedback...

5.7 – Testes de Recuperação

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Muitos sistemas de computador devem se recuperar de falhas e retomar o processamento em pouco ou nenhum tempo de parada. Em alguns casos, um sistema tem de ser tolerante a falhas; ou seja, falhas no processamento não devem causar a paralisação total do sistema. Em outros casos, uma falha no sistema deve ser corrigida dentro de um determinado período de tempo, caso contrário, poderão ocorrer sérios prejuízos financeiros.

O teste de recuperação é um teste do sistema que força o software a falhar de várias formas e verifica se a recuperação é executada corretamente. Se a recuperação for automática (executada pelo próprio sistema), a reinicialização, os mecanismos de verificação, recuperação de dados e reinício são avaliados quanto à correção. Se a recuperação requer intervenção humana, o tempo médio de reparo (MTTR[2]) é avaliado para determinar se está dentro dos limites aceitáveis.

Esses testes validam a capacidade e qualidade da recuperação do software após crashes, falhas de hardware ou outros problemas catastróficos. Em outras palavras, os testes de recuperação forçam o software a falhar de várias formas e verificam se a recuperação foi adequada, podendo ocorrer manual ou automática. Galera, aqui não tem muito segredo... vamos partir direto para as perguntas sobre esse tema.

5.8 – Testes de Compatibilidade

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

[2] Mean Time To Repair

Diferentes computadores, dispositivos de imagem, sistemas operacionais, navegadores e velocidades de conexão de rede podem ter influência significativa na operação de software. Cada configuração de computador pode resultar em diferenças nas velocidades de processamento no lado do cliente, resoluções de tela e velocidades de conexão. Excentricidades de sistemas operacionais podem causar problemas no processamento da aplicação de software.

Diferentes navegadores, às vezes, produzem resultados ligeiramente diferentes. **Em alguns casos, pequenos problemas de compatibilidade não apresentam dificuldades significativas, mas – em outros – podem ser encontrados erros graves.** Por exemplo, velocidades de download podem se tornar inaceitáveis, a falta de um plug-in necessário pode tornar o conteúdo indisponível, diferenças entre navegadores podem mudar significativamente o layout da página, etc.

O teste de compatibilidade procura descobrir esses problemas antes que a aplicação de software fique disponível. A finalidade desses testes é descobrir erros ou problemas de execução que podem ser atribuídos a diferenças em configuração. Em suma, eles validam a capacidade do software de ser executado em um ambiente particular de hardware, software, sistema operacional, rede, entre outros – podendo ser automático ou manual.

5.9 – Testes de Segurança

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Qualquer sistema de computador que trabalhe com informações sensíveis ou que cause ações que podem inadequadamente prejudicar (ou beneficiar) indivíduos, é um alvo para acesso impróprio ou ilegal. **As invasões abrangem uma ampla gama de atividades: hackers que tentam invadir sistemas por diversão, funcionários desgostosos que tentam invadir por vingança, indivíduos desonestos que tentam invadir para obter ganhos pessoais ilícitos.**

O teste de segurança – teste de ameaça ou invasão – tenta verificar se os mecanismos de proteção incorporados ao sistema vão de fato protegê-lo contra acesso indevido. Com tempo e recursos suficientes, um bom teste de segurança finalmente conseguirá invadir o sistema. **O papel do criador do sistema é tornar o custo da invasão maior do que o valor das informações que poderiam ser obtidas.** Durante o teste, o testador faz o papel do indivíduo que quer invadir!

O testador pode tentar obter senhas por meios externos; pode atacar o sistema com software personalizado projetado para romper defesas; pode sobrecarregar o sistema, o sistema pode assim recusar serviço a outros; pode causar erros no sistema propositadamente, esperando poder invadir durante a recuperação; pode examinar dados que não estão em segurança, tentando encontrar a chave para a entrada no sistema.

QUESTÕES COMENTADAS

1. **(FCC – 2014 – TRT/1)** Considerando o teste de software, há o chamado teste de unidade, que consiste em testar:

   a) o software completo, incluindo todos os seus componentes ou módulos, no ambiente de testes.
   b) o funcionamento dos compiladores que estiverem sendo utilizados no desenvolvimento do software.
   c) individualmente, componentes ou módulos de software que, posteriormente devem ser testados de maneira integrada.
   d) o software completo em seu ambiente final de operação, já com o hardware base do projeto.
   e) apenas componentes ou módulos de software cujo código fonte tenha mais de 100 linhas.

Comentários:

(a) *Software completo? Incluindo todos os seus componentes ou módulos? No ambiente de testes?* Isso é um teste alfa; (b) Testes de Unidade não têm nenhuma relação com funcionamento de compiladores; (c) Correto, testam-se componentes e módulos individualmente e, posteriormente, testa-se em conjunto no teste de integração; (d) *Software completo? Em seu ambiente final de operação? Com o hardware base do projeto?* Isso é um teste beta! (e) Esse item não faz o menor sentido! Não é porque é um teste de unidade que ele é pequeno – não há relação com a quantidade de linhas.

Gabarito: Letra C

---

2. **(FCC - 2015 – TCM/GO)** Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da área de TI indicou a seguinte estratégia convencional para testes de um sistema que está sendo desenvolvido:

   I. Para cada componente ou módulo, testar a interface, a estrutura de dados local, os caminhos independentes ao longo da estrutura de controle e as condições-limite para garantir que a informação flui adequadamente para dentro e para fora do módulo, que todos os comandos tenham sido executados e que todos os caminhos de manipulação de erros sejam testados.

   II. Aplicar uma abordagem incremental de testes para a construção da arquitetura do sistema, de forma que os módulos testados sejam integrados a partir do módulo de controle principal e os testes sejam conduzidos à medida que cada componente é inserido.

O Auditor indicou em I e II, respectivamente, os testes de:

a) caixa branca e de caixa preta, que são suficientes para validar todo o sistema.

b) unidade e de integração; na sequência, indicou os testes de validação e de sistema que são adequados para validar todo o sistema.

c) unidade e de interoperabilidade; na sequência, indicou os testes de caixa branca e de caixa preta que são adequados para validar todo o sistema.

d) carga e de desempenho; na sequência, indicou os testes de usabilidade e interoperabilidade que são adequados para validar todo o sistema.

e) caixa preta e de caixa branca, que são suficientes para validar todo o sistema.

Comentários:

Testar interface de módulos é característica do teste de unidade; e aplicar uma abordagem incremental para a construção da arquitetura do sistema é característica do teste de integração. Além disso, os testes de validação e sistema são realmente adequados para validar todo sistema.

Gabarito: Letra B

---

3. **(FCC - 2015 – TCM/GO)** Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da Área de TI recebeu a tarefa de identificar testes que sejam capazes de verificar:

– a validade funcional do sistema;
– o comportamento e o desempenho do sistema;
– quais classes de entrada vão constituir bons casos de teste;
– se o sistema é sensível a certos valores de entrada;
– quais taxas e volumes de dados o sistema pode tolerar;
– que efeito combinações específicas de dados terão na operação do sistema.

A indicação correta do Auditor é utilizar:

a) testes de caixa branca.
b) mais de um tipo de teste, pois não há um único tipo de teste capaz de avaliar todas estas situações.
c) um tipo diferente de teste para cada uma das situações elencadas.
d) testes de caixa preta.
e) testes de desempenho para os 2 primeiros e de carga para os demais.

Comentários:

Apenas o teste caixa-preta já seria suficiente, porque ele cumpre todos os requisitos! Não é necessário conhecer a estrutura interna para realizar nenhum desses testes. Além disso, não faz sentido fazer testes de desempenho para verificar a validade funcional do sistema – testes de desempenho tratam de requisitos não-funcionais.

Gabarito: Letra D

---

4. **(FGV - 2016 – IBGE)** Os testes de aceitação são muitas vezes a última etapa de testes antes de implantar o software em produção. Seu objetivo maior é verificar se o software está apto para utilização por parte dos usuários finais, de acordo com os requisitos de implementação definidos. Há três estratégias de implementação de testes de aceitação: a aceitação formal, a aceitação informal (ou teste alfa) e o teste beta.

   Com relação às três estratégias de implementação dos testes de aceitação, é correto afirmar que:

   a) o teste de aceitação informal, ou teste alfa, é conduzido nas instalações do usuário final, geralmente sem a presença do desenvolvedor;

   b) o teste beta é conduzido na instalação do desenvolvedor por um grupo representativo de usuários finais;

   c) o teste de aceitação formal utiliza todo o conjunto de casos de teste aplicados durante o teste do sistema, para procurar novos problemas;

   d) o teste beta é focado na busca de defeitos e seu progresso é facilmente medido;

   e) o teste de aceitação formal pode ser realizado de forma automatizada.

Comentários:

(a) Errado. Teste Alfa é conduzido nas instalações do desenvolvedor; (b) Errado. Teste Beta é conduzido nas instalações do usuário final; (c) Errado. Conforme vimos em aula, ele costuma ser uma extensão do Teste de Sistema. No entanto, trata-se de um subconjunto dos testes que foram realizados no Teste de Sistema; (d) Errado. Teste Beta é mais um teste de aceitação de um produto de software do que um teste focado em buscar defeitos, além disso não é fácil medir seu progresso; (e) Certo. Ele pode – sim – ser realizado de forma automatizada.

Gabarito: Letra E

---

5. **(FGV - 2016 – IBGE)** Trata-se de um teste que desconhece o conteúdo do código fonte. Nesse teste o componente testado é tratado como uma caixa preta: são fornecidos dados de

entrada e o resultado comparado com aquele esperado e previamente conhecido. Além disso, esse teste pode ser aplicado em diversas fases de teste. A questão retrata características do teste:

a) funcional;
b) de integração;
c) de desempenho;
d) de carga;
e) unitário.

Comentários:

Pela primeira frase dessa questão, já era possível descobrir: nos testes funcionais, ignoram-se detalhes de implementação – também chamados de testes caixa-preta.

Gabarito: Letra A

---

6. **(FGV - 2014 – PROCEM/PA)** A verificação dinâmica está baseada nas três dimensões de testes, listadas a seguir: tipos de teste, técnicas de teste e níveis de teste. Assinale a opção que apresenta somente itens da dimensão tipos de teste.

a) Teste de Aceitação – Teste de Regressão – Teste Estrutural
b) Teste de Funcionalidade – Teste de Desempenho – Teste de Unidade
c) Teste de Interface – Teste de Carga – Teste de Segurança
d) Teste Funcional – Teste de Volume – Teste de Sistema
e) Teste de Usabilidade – Teste de Funcionalidade – Teste de Integração

Comentários:

(a) Errado. Nível de Teste; Tipo de Teste; Técnica de Teste;
(b) Errado. Tipo de Teste; Tipo de Teste; Nível de Teste;
(c) Correto. Tipo de Teste; Tipo de Teste; Tipo de Teste;
(d) Errado. Técnica de Teste; Tipo de Teste; Nível de Teste;
(e) Errado. Tipo de Teste; Tipo de Teste; Nível de Teste;

Gabarito: Letra C

---

7. **(FGV - 2017 – ALERJ)** A atividade de teste de software contribui para revelar defeitos latentes nos programas. Em relação às técnicas de testes de software, é correto afirmar que:

a) testes de caixa branca têm por objetivo testar o código-fonte, testar cada linha de código possível, testar os fluxos básicos e os alternativos;

b) testes de regressão têm por objetivo verificar se o sistema se mantém funcionando de maneira satisfatória após longos e intensos períodos de uso;

c) todas as declarações internas do programa devem ser testadas pelo menos uma vez durante os testes funcionais;

d) testes de unidade se preocupam em exercitar o sistema além de sua carga máxima de projeto, até que ele falhe;

e) testes de usabilidade verificam se o software instala como planejado, em diferentes hardwares e sob diferentes condições.

Comentários:

(a) Correto, em testes caixa-branca, teoricamente você deve testar todo caminho possível através do código – na prática, isso é meio inviável. (b) Errado, isso é um teste de estabilidade; (c) Errado, em testes funcionais não se testam declarações internas; (d) Errado, isso é um teste de estresse; (e) Errado, isso é um teste de instalação.

Gabarito: Letra A

---

8. **(CESGRANRIO – 2016 – IBGE)** O sistema que controla as reservas dos clientes de uma rede hoteleira funciona apenas na Web. Entretanto, há uma demanda crescente para que a empresa disponibilize um aplicativo para smartphones. Para oferecer um aplicativo no menor prazo possível, a gerência de TI estabeleceu duas exigências: a primeira é que o novo sistema deve reutilizar ao máximo os módulos atualmente empregados, e a segunda é que a equipe de desenvolvimento deve garantir que as modificações a serem feitas não introduzirão defeitos inexistentes no sistema atual, além de continuar a atender a todos os requisitos anteriormente definidos.

O tipo de teste que deve ser empregado para que a equipe de desenvolvimento atenda à segunda exigência é denominado teste de:

a) estresse
b) volume
c) usabilidade
d) regressão
e) configuração

Comentários:

A finalidade do teste de regressão é justamente aplicar testes em versões mais recentes do software para garantir que não surgiram novos defeitos em componentes já analisados. Ou ainda

em situações como a descrita no enunciado em que novos componentes desenvolvidos para a plataforma mobile e/ou alterações em componentes existentes não criem novos defeitos em componentes inalterados e já testados. Caso testes comecem a não passar, então considera-se que o sistema regrediu.

Gabarito: Letra D

---

9. (CESGRANRIO – 2014 – Banco da Amazônia) Um tipo de teste de validação possui as seguintes características:

- Realizado na instalação dos desenvolvedores.
- Conduzido em um ambiente controlado.
- Conta com a participação de usuários e desenvolvedores.

Esse tipo de teste é chamado de:

a) Teste alfa
b) Teste beta
c) Teste de estresse
d) Teste de regressão
e) Teste de desempenho

Comentários:

O teste em que os usuários finais do software o testam sob as observações dos desenvolvedores nas instalações destes – em um ambiente controlado – é o Teste Alfa.

Gabarito: Letra A

---

10. (CESGRANRIO – 2014 – IBGE) Antes de lançar seu próximo produto, uma empresa de desenvolvimento de software costuma convidar seus principais clientes para testar uma versão "quase final". Esses clientes são convidados ao local de desenvolvimento e são observados enquanto utilizam o software e registram erros e problemas no uso.

Essa estratégia de teste em um ambiente controlado é conhecida como teste:

a) alfa
b) beta
c) de stress
d) de integração
e) de aceitação do cliente

Comentários:

Local de desenvolvimento? Observados enquanto utilizam o software? Ambiente controlado? Tudo isso nos remete a testes alfa.

Gabarito: Letra A

---

11. (CESGRANRIO – 2014 – IBGE) No ciclo de desenvolvimento de sistemas, os testes são de suma importância e podem, dependendo do porte do sistema, ser bastante complexos, exigindo que seu planejamento e realização sejam divididos em fases. Em uma dessas fases, os testes são realizados por um grupo restrito de usuários finais do sistema, que simulam operações de rotina do sistema, de modo a verificar se seu comportamento está de acordo com o solicitado. Essa fase é denominada teste de:

a) integração
b) unidade
c) sistema
d) operação
e) aceitação

Comentários:

O teste que verifica se o sistema cumpre o que foi solicitado é o teste de aceitação ou validação!

Gabarito: Letra E

---

12. (CESGRANRIO – 2014 – IBGE) Preocupado com os constantes erros nos sistemas entregues aos usuários, um analista de desenvolvimento resolveu realizar testes conforme o modelo V. A correspondência da validação dos modelos por fases do processo de software, de acordo com esse modelo, está representada em:

a) Teste unitário → Modelagem de requisitos
b) Teste de sistema → Desenvolvimento de componentes
c) Teste de aceitação → Geração de código
d) Teste de integração → Arquitetura do sistema
e) Teste de caixa preta → Desenho das estruturas de dados

Comentários:

O **Teste de Integração** é uma técnica sistemática para construir a **arquitetura de software** ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces. O objetivo é construir uma estrutura de programa determinada pelo projeto a partir de componentes testados em nível de unidade.

Gabarito: Letra D

---

13. (CESGRANRIO - 2018 – TRANSPETRO) Entre as técnicas de teste de software, aquela que gera versões levemente modificadas de um programa sob teste e exercita tanto o programa original quanto os programas modificados, procurando diferenças entre essas formas, é conhecida como testes:

a) aleatórios
b) exploratórios
c) de mutação
d) de perfil operacional
e) baseados em fluxo de controle

Comentários:

(a) Errado. Teste Aleatório é o chamado Teste Macaco em que as entradas são aleatórias; (b) Errado. Teste Exploratório é um teste em que o analista de teste aprende e testa no mesmo momento, explorando o sistema para aprender durante o teste; (c) Teste de Mutação é tipo de teste onde esses programas levemente modificados servem para identificar testes fracos; (d) Errado. Teste de Perfil Operacional é um teste de qualidade realizado antes da release; (e) Errado. Testes baseados em Fluxo de Controle é um teste caixa-branca que exige que se conheça as estruturas internas para forças todos os estados que geram saídas diferentes.

Gabarito: Letra C

---

14. (CESPE - 2015 – TJDFT) As atividades de validação incluem os testes unitários e os de aceitação.

Comentários:

Testes Unitários não são atividades de validação, mas de verificação! À princípio, a banca deu o item como correto, mas depois o anulou sob a justificativa de que "*a redação do item prejudicou seu julgamento objetivo*". Galera, na minha opinião, não há nada de errado com a redação – o item está completamente errado.

Gabarito: ANULADO

---

15. (CESPE – 2017 – TRE/BA) Durante o planejamento de um projeto e a elaboração dos casos de uso, foram incluídos diversos componentes para cálculos de tributos e da quantidade de recursos orçamentários alocados. De acordo com os níveis de testes existentes, o resultado de um dos componentes em questão poderá ser validado por meio do teste:

a) unitário.

b) de sistema.
c) de aceitação.
d) de codificação.
e) de integração.

Comentários:

Questão que você não pode errar, porque todo mundo acertará! A questão fala em diversos componentes para cálculos e fala em níveis de teste (importante saber todos os níveis de teste), logo – para testar um componente – utilizam-se testes unitários.

Gabarito: Letra A

---

16. (CESPE – 2017 – TRE/BA) O gestor de um órgão organizador de concursos públicos pretende oferecer condições para que mais de um milhão de candidatos inscritos em determinado evento possa obter o gabarito das provas a partir do acesso ao seu sistema eletrônico. Nessa situação, para verificar se o sistema eletrônico suportará uma quantidade grande de acessos simultâneos, a equipe de TI do órgão, ao preparar o ambiente de acesso eletrônico, deverá realizar o teste:

a) de estresse.
b) unitário.
c) integrado.
d) de sistema.
e) de regressão.

Comentários:

Essa questão é bastante mal escrita, visto que uma "*grande quantidade*" é muito subjetivo! Sabemos que testes de estresse testam cargas acima das especificadas/esperadas até descobrir em que ponto o sistema começa a falhar. Logo, para mim, essa questão não tem resposta – porém, a primeira opção é a menos errada.

Gabarito: Letra A

---

17. (CESPE - 2018 – STM) Testes de regressão servem ao propósito de verificar se o sistema pode operar na carga necessária, fazendo-a regredir constantemente até que o comportamento de falha do sistema seja testado ou que defeitos sejam identificados.

Comentários:

Carga? Nope! Testes de Regressão são executados para garantir que uma funcionalidade ou parte de um software já testado continue funcionando após o mesmo sofrer alguma implementação ou manutenção.

Gabarito: Errado

---

18. (CESPE - 2018 – STM) Em testes de integração, a estratégia de integração bottom-up integrará componentes de infraestrutura que fornecem serviços comuns, adicionando a eles componentes funcionais; para testar uma nova característica, pode ser necessário integrar componentes diferentes.

Comentários:

O que é Bottom-Up? Eu construo, integro e testo componentes de infraestrutura que fornecem serviços comuns a vários outros componentes; são mais genéricos, ou seja, é aquela classe mais independente que presta serviço para outras e depois componentes funcionais em que eu testo componentes que realizam funcionalidades específicas, que são mais dependentes, que precisam de outras classes – então é do mais geral para o mais específico (no sentido de funcionalidades) e do menor para o maior (no sentido de tamanho). A abordagem top-down é o contrário, eu começo testando o sistema até chegar aos módulos, ou seja, do mais específico para o mais geral (no sentido de funcionalidade). Portanto, a questão está perfeita!

Gabarito: Correto

---

19. (FUMARC - 2014 – AL/MG) Tipo de teste que consiste em aplicar em cada nova versão de um software todos os testes que já foram aplicados nas versões anteriores, possibilitando a identificação dos impactos das implementações da nova versão em funcionalidades que já foram testadas anteriormente e não foram modificadas, é denominado:

a) Teste Caixa-branca.
b) Teste Caixa-preta.
c) Teste de Regressão.
d) Teste Unitário.

Comentários:

Aplicar em cada nova versão de um software todos os testes que já foram aplicados nas versões anteriores, possibilitando a identificação dos impactos das implementações da nova versão em funcionalidades que já foram testadas anteriormente e não foram modificadas, é denominado teste de regressão.

Gabarito: Letra C

---

20. **(IF-PE - 2016 – IF-PE)** Em relação aos Testes na Engenharia de Software, qual é o que se refere ao reteste de uma unidade, integração ou sistema, após uma modificação, a fim de verificar se a mudança não introduziu novas falhas?

a) Teste de integração.
b) Teste de regressão.
c) Teste alfa.
d) Teste beta.
e) Teste funcional.

Comentários:

Restes de uma unidade, integração ou sistema, após uma modificação, a fim de verificar se a mudança não introduziu novas falhas é característica do teste de regressão.

Gabarito: Letra B

---

21. **(FAURGS - 2018 – BANRISUL)** _____ verifica novamente os casos de teste aprovados em versões prévias do software e assim protege contra alterações indesejadas. Realiza-se durante a manutenção, para mostrar que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados funcionam como o esperado e que os requisitos anteriormente testados continuam válidos.

Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Teste de regressão
b) Teste de sistema
c) Inspeção
d) Refatoração
e) Teste de integração

Comentários:

Quem verifica novamente os casos de teste aprovados em versões prévias do software e assim protege contra alterações indesejadas para mostrar que as modificações efetuadas estão corretas é o teste de regressão.

Gabarito: Letra A

---

22. **(UFLA - 2018 – UFLA)** Sobre teste de regressão de software, são verdadeiras as afirmativas abaixo, EXCETO:

a) Consiste na reexecução de testes previamente realizados no software para garantir que modificações ou novas funcionalidades não introduzam comportamentos indesejáveis ou erros adicionais.
b) É sempre prático e viável executar o teste de regressão após cada modificação do software.
c) Embora normalmente seja conduzido de forma automática, o teste de regressão pode também ser conduzido manualmente.
d) Teste de regressão pode ser configurado para execução em servidores de integração contínua.

Comentários:

Todos os itens estão perfeitos, exceto o segundo – nem sempre é prático executar o teste de regressão a cada modificação. À medida que o teste de integração progride, o número de testes de regressão pode crescer muito. Dessa forma, o conjunto de testes de regressão deve ser projetado de forma a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. É impraticável e ineficiente reexecutar todos os testes para todas as funções do programa quando ocorre uma alteração.

Gabarito: Letra B

---

23. (CESPE – 2017 – TRT/CE) A respeito de engenharia de software, assinale a opção correta.

a) A finalidade dos testes de segurança é garantir que o sistema se recupere de uma falha e esteja apto a retomar o processamento em um prazo preestabelecido.

b) Efetuar testes de regressão consiste em reexecutar testes já finalizados para garantir que eventuais alterações não tenham impactado funções que antes funcionavam corretamente.

c) Os testes de integração ascendentes são caracterizados pelo fato de a sua realização ocorrer conforme o desenvolvimento dos módulos.

d) Na etapa de desenvolvimento de um software, os testes de validação e de integração são executados simultaneamente, para identificar inconsistências antes da entrega final.

Comentários:

(a) Errado, esse é o teste de recuperação; (b) Correto, é a definição impecável de teste de regressão; (c) Errado, testes de integração ascendentes são caracterizados pela integração dos módulos de baixo nível primeiro; (d) Errado, testes de validação são posteriores.

Gabarito: Letra B

---

24. **(FCM– 2016 – IF Farroupilha/RS)** O teste de software pode ser realizado de diversas formas. Mesmo assim, existem técnicas que podem ser utilizadas para encontrar falhas no software. Analise as afirmativas abaixo:

I - O teste de regressão tem por finalidade repetir o teste em um programa já testado depois de haver uma modificação.

II - O teste de desempenho tem por finalidade elaborar casos de teste que possam subverter as verificações de segurança do programa.

III - O teste de caixa branca trabalha diretamente sobre o código fonte do componente de software.

IV - O teste de caixa preta trabalha diretamente sobre o código fonte do componente de software.

Estão corretas as afirmativas:

a) I e II.
b) I e III.
c) II e IV.
d) II e III.
e) I, II, III e IV.

Comentários:

(I) Correto, esse teste realmente tem por finalidade repetir o teste em um programa já testado depois de haver uma modificação; (II) Errado, esse é o teste de segurança; (III) Correto, ele verifica os componentes internos de um software; (IV) Errado, esse é o teste caixa-branca – testes caixa-preta não possuem acesso ao código-fonte de componentes de software.

Gabarito: Letra B

---

25. **(CESPE – 2018 – BNB)** Uma equipe de desenvolvimento de softwares pretendia realizar testes de forma incremental durante o desenvolvimento de um programa, a fim de verificar se mudanças no programa não haviam nele introduzido novos bugs; para isso, foram sugeridos os testes unitários e de regressão. Nessa situação, será correto utilizar os testes unitários, mas não os testes de regressão, pois esses últimos não visam verificar novos bugs, mas sim, tão somente, avaliar as funcionalidades do sistema.

Comentários:

Pelo contrário, esse é o principal teste responsável por verificar novos bugs inseridos no sistema.

Gabarito: Errado

LISTA DE QUESTÕES

1. **(FCC – 2014 – TRT/1)** Considerando o teste de software, há o chamado teste de unidade, que consiste em testar:

   a) o software completo, incluindo todos os seus componentes ou módulos, no ambiente de testes.
   b) o funcionamento dos compiladores que estiverem sendo utilizados no desenvolvimento do software.
   c) individualmente, componentes ou módulos de software que, posteriormente devem ser testados de maneira integrada.
   d) o software completo em seu ambiente final de operação, já com o hardware base do projeto.
   e) apenas componentes ou módulos de software cujo código fonte tenha mais de 100 linhas.

2. **(FCC - 2015 – TCM/GO)** Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da área de TI indicou a seguinte estratégia convencional para testes de um sistema que está sendo desenvolvido:

   I. Para cada componente ou módulo, testar a interface, a estrutura de dados local, os caminhos independentes ao longo da estrutura de controle e as condições-limite para garantir que a informação flui adequadamente para dentro e para fora do módulo, que todos os comandos tenham sido executados e que todos os caminhos de manipulação de erros sejam testados.

   II. Aplicar uma abordagem incremental de testes para a construção da arquitetura do sistema, de forma que os módulos testados sejam integrados a partir do módulo de controle principal e os testes sejam conduzidos à medida que cada componente é inserido.

O Auditor indicou em I e II, respectivamente, os testes de:

   a) caixa branca e de caixa preta, que são suficientes para validar todo o sistema.

   b) unidade e de integração; na sequência, indicou os testes de validação e de sistema que são adequados para validar todo o sistema.

   c) unidade e de interoperabilidade; na sequência, indicou os testes de caixa branca e de caixa preta que são adequados para validar todo o sistema.

   d) carga e de desempenho; na sequência, indicou os testes de usabilidade e interoperabilidade que são adequados para validar todo o sistema.

e) caixa preta e de caixa branca, que são suficientes para validar todo o sistema.

3. **(FCC - 2015 – TCM/GO)** Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da Área de TI recebeu a tarefa de identificar testes que sejam capazes de verificar:

– a validade funcional do sistema;
– o comportamento e o desempenho do sistema;
– quais classes de entrada vão constituir bons casos de teste;
– se o sistema é sensível a certos valores de entrada;
– quais taxas e volumes de dados o sistema pode tolerar;
– que efeito combinações específicas de dados terão na operação do sistema.

A indicação correta do Auditor é utilizar:

a) testes de caixa branca.
b) mais de um tipo de teste, pois não há um único tipo de teste capaz de avaliar todas estas situações.
c) um tipo diferente de teste para cada uma das situações elencadas.
d) testes de caixa preta.
e) testes de desempenho para os 2 primeiros e de carga para os demais.

4. **(FGV - 2016 – IBGE)** Os testes de aceitação são muitas vezes a última etapa de testes antes de implantar o software em produção. Seu objetivo maior é verificar se o software está apto para utilização por parte dos usuários finais, de acordo com os requisitos de implementação definidos. Há três estratégias de implementação de testes de aceitação: a aceitação formal, a aceitação informal (ou teste alfa) e o teste beta.

Com relação às três estratégias de implementação dos testes de aceitação, é correto afirmar que:

a) o teste de aceitação informal, ou teste alfa, é conduzido nas instalações do usuário final, geralmente sem a presença do desenvolvedor;

b) o teste beta é conduzido na instalação do desenvolvedor por um grupo representativo de usuários finais;

c) o teste de aceitação formal utiliza todo o conjunto de casos de teste aplicados durante o teste do sistema, para procurar novos problemas;

d) o teste beta é focado na busca de defeitos e seu progresso é facilmente medido;

e) o teste de aceitação formal pode ser realizado de forma automatizada.

5. **(FGV - 2016 – IBGE)** Trata-se de um teste que desconhece o conteúdo do código fonte. Nesse teste o componente testado é tratado como uma caixa preta: são fornecidos dados de entrada e o resultado comparado com aquele esperado e previamente conhecido. Além disso, esse teste pode ser aplicado em diversas fases de teste. A questão retrata características do teste:

a) funcional;
b) de integração;
c) de desempenho;
d) de carga;
e) unitário.

6. **(FGV - 2014 – PROCEM/PA)** A verificação dinâmica está baseada nas três dimensões de testes, listadas a seguir: tipos de teste, técnicas de teste e níveis de teste. Assinale a opção que apresenta somente itens da dimensão tipos de teste.

a) Teste de Aceitação – Teste de Regressão – Teste Estrutural
b) Teste de Funcionalidade – Teste de Desempenho – Teste de Unidade
c) Teste de Interface – Teste de Carga – Teste de Segurança
d) Teste Funcional – Teste de Volume – Teste de Sistema
e) Teste de Usabilidade – Teste de Funcionalidade – Teste de Integração

7. **(FGV - 2017 – ALERJ)** A atividade de teste de software contribui para revelar defeitos latentes nos programas. Em relação às técnicas de testes de software, é correto afirmar que:

a) testes de caixa branca têm por objetivo testar o código-fonte, testar cada linha de código possível, testar os fluxos básicos e os alternativos;

b) testes de regressão têm por objetivo verificar se o sistema se mantém funcionando de maneira satisfatória após longos e intensos períodos de uso;

c) todas as declarações internas do programa devem ser testadas pelo menos uma vez durante os testes funcionais;

d) testes de unidade se preocupam em exercitar o sistema além de sua carga máxima de projeto, até que ele falhe;

e) testes de usabilidade verificam se o software instala como planejado, em diferentes hardwares e sob diferentes condições.

8. **(CESGRANRIO – 2016 – IBGE)** O sistema que controla as reservas dos clientes de uma rede hoteleira funciona apenas na Web. Entretanto, há uma demanda crescente para que a empresa disponibilize um aplicativo para smartphones. Para oferecer um aplicativo no menor prazo possível, a gerência de TI estabeleceu duas exigências: a primeira é que o novo sistema

deve reutilizar ao máximo os módulos atualmente empregados, e a segunda é que a equipe de desenvolvimento deve garantir que as modificações a serem feitas não introduzirão defeitos inexistentes no sistema atual, além de continuar a atender a todos os requisitos anteriormente definidos.

O tipo de teste que deve ser empregado para que a equipe de desenvolvimento atenda à segunda exigência é denominado teste de:

a) estresse
b) volume
c) usabilidade
d) regressão
e) configuração

9. **(CESGRANRIO – 2014 – Banco da Amazônia)** Um tipo de teste de validação possui as seguintes características:

- Realizado na instalação dos desenvolvedores.
- Conduzido em um ambiente controlado.
- Conta com a participação de usuários e desenvolvedores.

Esse tipo de teste é chamado de:

a) Teste alfa
b) Teste beta
c) Teste de estresse
d) Teste de regressão
e) Teste de desempenho

10. **(CESGRANRIO – 2014 – IBGE)** Antes de lançar seu próximo produto, uma empresa de desenvolvimento de software costuma convidar seus principais clientes para testar uma versão "quase final". Esses clientes são convidados ao local de desenvolvimento e são observados enquanto utilizam o software e registram erros e problemas no uso.

Essa estratégia de teste em um ambiente controlado é conhecida como teste:

a) alfa
b) beta
c) de stress
d) de integração
e) de aceitação do cliente

11. **(CESGRANRIO – 2014 – IBGE)** No ciclo de desenvolvimento de sistemas, os testes são de suma importância e podem, dependendo do porte do sistema, ser bastante complexos,

exigindo que seu planejamento e realização sejam divididos em fases. Em uma dessas fases, os testes são realizados por um grupo restrito de usuários finais do sistema, que simulam operações de rotina do sistema, de modo a verificar se seu comportamento está de acordo com o solicitado. Essa fase é denominada teste de:

a) integração
b) unidade
c) sistema
d) operação
e) aceitação

12. (CESGRANRIO – 2014 – IBGE) Preocupado com os constantes erros nos sistemas entregues aos usuários, um analista de desenvolvimento resolveu realizar testes conforme o modelo V. A correspondência da validação dos modelos por fases do processo de software, de acordo com esse modelo, está representada em:

a) Teste unitário → Modelagem de requisitos
b) Teste de sistema → Desenvolvimento de componentes
c) Teste de aceitação → Geração de código
d) Teste de integração → Arquitetura do sistema
e) Teste de caixa preta → Desenho das estruturas de dados

13. (CESGRANRIO - 2018 – TRANSPETRO) Entre as técnicas de teste de software, aquela que gera versões levemente modificadas de um programa sob teste e exercita tanto o programa original quanto os programas modificados, procurando diferenças entre essas formas, é conhecida como testes:

a) aleatórios
b) exploratórios
c) de mutação
d) de perfil operacional
e) baseados em fluxo de controle

14. (CESPE - 2015 – TJDFT) As atividades de validação incluem os testes unitários e os de aceitação.

15. (CESPE – 2017 – TRE/BA) Durante o planejamento de um projeto e a elaboração dos casos de uso, foram incluídos diversos componentes para cálculos de tributos e da quantidade de recursos orçamentários alocados. De acordo com os níveis de testes existentes, o resultado de um dos componentes em questão poderá ser validado por meio do teste:

a) unitário.
b) de sistema.
c) de aceitação.

d) de codificação.
e) de integração.

16. **(CESPE – 2017 – TRE/BA)** O gestor de um órgão organizador de concursos públicos pretende oferecer condições para que mais de um milhão de candidatos inscritos em determinado evento possa obter o gabarito das provas a partir do acesso ao seu sistema eletrônico. Nessa situação, para verificar se o sistema eletrônico suportará uma quantidade grande de acessos simultâneos, a equipe de TI do órgão, ao preparar o ambiente de acesso eletrônico, deverá realizar o teste:

a) de estresse.
b) unitário.
c) integrado.
d) de sistema.
e) de regressão.

17. **(CESPE - 2018 – STM)** Testes de regressão servem ao propósito de verificar se o sistema pode operar na carga necessária, fazendo-a regredir constantemente até que o comportamento de falha do sistema seja testado ou que defeitos sejam identificados.

18. **(CESPE - 2018 – STM)** Em testes de integração, a estratégia de integração bottom-up integrará componentes de infraestrutura que fornecem serviços comuns, adicionando a eles componentes funcionais; para testar uma nova característica, pode ser necessário integrar componentes diferentes.

19. **(FUMARC - 2014 – AL/MG)** Tipo de teste que consiste em aplicar em cada nova versão de um software todos os testes que já foram aplicados nas versões anteriores, possibilitando a identificação dos impactos das implementações da nova versão em funcionalidades que já foram testadas anteriormente e não foram modificadas, é denominado:

a) Teste Caixa-branca.
b) Teste Caixa-preta.
c) Teste de Regressão.
d) Teste Unitário.

20. **(IF-PE - 2016 – IF-PE)** Em relação aos Testes na Engenharia de Software, qual é o que se refere ao reteste de uma unidade, integração ou sistema, após uma modificação, a fim de verificar se a mudança não introduziu novas falhas?

a) Teste de integração.
b) Teste de regressão.
c) Teste alfa.
d) Teste beta.
e) Teste funcional.

21. **(FAURGS - 2018 – BANRISUL)** _____ verifica novamente os casos de teste aprovados em versões prévias do software e assim protege contra alterações indesejadas. Realiza-se durante a manutenção, para mostrar que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados funcionam como o esperado e que os requisitos anteriormente testados continuam válidos.

Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Teste de regressão
b) Teste de sistema
c) Inspeção
d) Refatoração
e) Teste de integração

22. **(UFLA - 2018 – UFLA)** Sobre teste de regressão de software, são verdadeiras as afirmativas abaixo, EXCETO:

a) Consiste na reexecução de testes previamente realizados no software para garantir que modificações ou novas funcionalidades não introduzam comportamentos indesejáveis ou erros adicionais.
b) É sempre prático e viável executar o teste de regressão após cada modificação do software.
c) Embora normalmente seja conduzido de forma automática, o teste de regressão pode também ser conduzido manualmente.
d) Teste de regressão pode ser configurado para execução em servidores de integração contínua.

23. **(CESPE – 2017 – TRT/CE)** A respeito de engenharia de software, assinale a opção correta.

a) A finalidade dos testes de segurança é garantir que o sistema se recupere de uma falha e esteja apto a retomar o processamento em um prazo preestabelecido.

b) Efetuar testes de regressão consiste em reexecutar testes já finalizados para garantir que eventuais alterações não tenham impactado funções que antes funcionavam corretamente.

c) Os testes de integração ascendentes são caracterizados pelo fato de a sua realização ocorrer conforme o desenvolvimento dos módulos.

d) Na etapa de desenvolvimento de um software, os testes de validação e de integração são executados simultaneamente, para identificar inconsistências antes da entrega final.

24. **(FCM– 2016 – IF Farroupilha/RS)** O teste de software pode ser realizado de diversas formas. Mesmo assim, existem técnicas que podem ser utilizadas para encontrar falhas no software. Analise as afirmativas abaixo:

I - O teste de regressão tem por finalidade repetir o teste em um programa já testado depois de haver uma modificação.

II - O teste de desempenho tem por finalidade elaborar casos de teste que possam subverter as verificações de segurança do programa.

III - O teste de caixa branca trabalha diretamente sobre o código fonte do componente de software.

IV - O teste de caixa preta trabalha diretamente sobre o código fonte do componente de software.

Estão corretas as afirmativas:

a) I e II.
b) I e III.
c) II e IV.
d) II e III.
e) I, II, III e IV.

25. **(CESPE – 2018 – BNB)** Uma equipe de desenvolvimento de softwares pretendia realizar testes de forma incremental durante o desenvolvimento de um programa, a fim de verificar se mudanças no programa não haviam nele introduzido novos bugs; para isso, foram sugeridos os testes unitários e de regressão. Nessa situação, será correto utilizar os testes unitários, mas não os testes de regressão, pois esses últimos não visam verificar novos bugs, mas sim, tão somente, avaliar as funcionalidades do sistema.

GABARITO

1. LETRA C
2. LETRA B
3. LETRA D
4. LETRA E
5. LETRA A
6. LETRA C
7. LETRA A
8. LETRA D
9. LETRA A
10. LETRA A
11. LETRA E
12. LETRA D
13. LETRA C
14. ANULADO
15. LETRA A
16. LETRA A
17. ERRADO
18. CORRETO
19. LETRA C
20. LETRA B
21. LETRA A
22. LETRA B
23. LETRA B
24. LETRA B
25. ERRADO

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.